TITLE: M-Pesa Spending Predictor using Machine Learning

Objective:

To develop a web-based machine learning application that analyzes M-Pesa transaction statements and predicts future spending based on user input.

Project Description:

The M-Pesa Spending Predictor is a machine learning-powered web application built with Flask and XGBoost. It allows users to input details of their M-Pesa transactions and get predictions of their expected spending. The system is trained using historical transaction data and uses a combination of preprocessing, encoding, scaling, and regression modeling techniques.

For code review open GitHub account: mpesa prediction

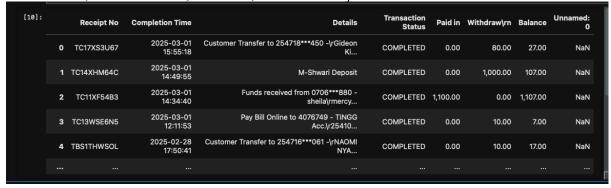
Tools & Technologies:

- 1. Python (Data processing, Model training)
- 2. Flask (API backend)
- 3. HTML/JavaScript (Frontend)
- 4. XGBoost (Regression model)
- 5. scikit-learn (Preprocessing, Evaluation)
- 6. pandas/NumPy (Data manipulation)
- 7. pickle (Model serialization)
- 8. LabelEncoder / StandardScaler
- 9. CORS / flask_restful

Data Used:

Original M-pesa statement (pdf format) is cleaned and anonymized from ['Receipt No', 'Completion Time', 'Details', 'Transaction Status',

'Paid in', 'Withdraw\rn', 'Balance', 'Unnamed: 0']

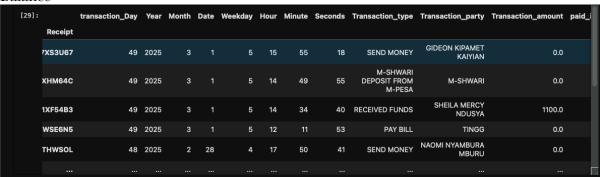


Го а

Cleaned M-Pesa transaction CSV with the following features:

- 1. Receipt
- 2. transaction_Day (number of transactions in a day)
- 3. Year
- 4. Month
- 5. Date

- 6. Weekday
- 7. Hour
- 8. Minute
- 9. Seconds
- 10. Transaction_type
- 11. Transaction_party
- 12. Transaction_amount
- 13. paid_in_or_Withdraw
- 14. Balance



Feature Engineering:

- 1. Extracted relevant time features from the date and time columns.
- 2. Filtered and encoded categorical variables using LabelEncoder.
- 3. Special handling of infrequent Transaction_party values.
- 4. Applied feature scaling using StandardScaler.

Model Training & Evaluation:

1. Used XGBRegressor with hyperparameter tuning via GridSearchCV.

Evaluated model using:

- 1. Root Mean Squared Error (RMSE) 40.19
- 2. Mean Absolute Error (MAE) 24.71
- 3. R2 Score 0.47

Flask API:

/Prediction - Accepts POST requests with transaction details and returns predicted spending. /Data - Returns historical transaction data in JSON format.

Frontend:

- 1. HTML form with fields for Transaction Amount, Type, Party, and Paid/Withdrawn.
- 2. JavaScript to submit data via Fetch API and render results.

Model Saving:

1. Serialized trained model, scaler, and encoder into .pkl files for use in the API.

Challenges & Solutions:

- 1. Handled unseen categorical values by mapping them to constant value eg balance 1000.
- 2. Addressed scaling issues by fitting the scaler only on training data.
- 3. Improved model performance via feature selection and parameter tuning.