

BUDGETWISE AI BASED FORECASTING TOOL





PROJECT AT A GLANCE

TEAM MEMBERS:

GUNDAPU DIVYA SREE

GUIDED BY(MENTOR):

SANGEETHA MAHALINGAM

PROJECT OVERVIEW

- >ABSTRACT
- >WHY THIS PROJECT?
- >OBJECTIVES
- >PROPOSED SOLUTION

DATASET

- >DATSET DETAILS
- >PARSING AND CLEANING
- >FEATURE PREPARATION
- >TRAIN-TEST SPLIT
- >DATA PREPARED FOR MODEL

MODEL DEVELOPMENT

- >MODEL SELECTION
- >MODEL TRAINING
- >STACKED ENSEMBLE ARCHITECTURE
- >MODEL EVALUATION AND COMPARISON

DEPLOYMENT

- >SYSTEM ARCHITECTURE
- >WEB APPLICATION FEATURES
- >MODEL INTEGRATION AND BACKEND LOGIC
- >HOSTING AND DEPLOYMENT

ABSTRACT

Financial planning and efficient budget management are essential for individuals and small businesses. To address these challenges, *BudgetWise AI* has been developed as an intelligent financial forecasting and budgeting system powered by Artificial Intelligence (AI) and Machine Learning (ML). The system analyzes historical financial data—such as income, expenses, and saving patterns—to accurately predict future expenditures and provide personalized budgeting insights. Unlike traditional budgeting applications that use static formulas, BudgetWise AI dynamically learns from user behavior and adapts its predictions using advanced ML models, including CatBoost, XGBoost, LightGBM, and a stacked ensemble for maximum accuracy. The platform also features an interactive Streamlit-based dashboard with visualizations such as category-wise spending charts, monthly trends, and budget comparisons.

Users can track expenses, manage recurring costs, set budgets, and receive AI-driven savings recommendations through an integrated financial advisor powered by the Gemini API. Secure authentication, encrypted data handling, and SQLite storage ensure user data privacy and reliability.

This project aims to provide a practical, AI-driven financial management assistant that empowers users to plan better, save more, and maintain long-term financial stability through intelligent automation.

WHY THIS PROJECT

- Managing personal finances has become challenging for students, individuals, and small businesses.
 - Existing budgeting apps only show past expenses and fail to predict future spending.
 - Users struggle with overspending because they cannot see upcoming financial patterns.
 - There is a lack of smart, AI-powered tools that provide forecasting, insights, and budget planning.
 - BudgetWise AI fills this gap by using Machine Learning to:
 - Analyze spending behavior
 - Predict monthly expenses
 - Offer personalized financial recommendations
 - Help users plan ahead rather than react later
- The project aims to make financial planning simple, intelligent, and accessible for everyone.

OBJECTIVES

To develop an AI-driven system capable of accurately forecasting future expenses and budgets.

To analyze and visualize user spending patterns using historical financial data.

To provide personalized financial planning through adaptive machine learning models.

To design an intuitive and user-friendly dashboard for insights, trends, and recommendations.

To ensure secure data handling and privacy through robust authentication and encryption.

To build a scalable architecture capable of supporting diverse financial datasets and multiple users.

PROPOSED SOLUTION

A full-stack, AI-powered financial management system built using Streamlit and SQLite.

Uses advanced ML models—CatBoost, XGBoost, LightGBM, and a Stacked Ensemble—to accurately forecast future expenses.

Learns from historical data (Category, Amount, Date features) to predict spending for the next 7–90 days.

Provides interactive dashboards with category-wise charts, monthly trends, and budget vs. actual insights.

Includes smart automation: recurring expense logging, anomaly detection, and AI financial advisor powered by Google Gemini.

Ensures secure data handling using SHA-256 authentication and a scalable modular architecture.

DATASET DETAILS

A publicly available personal finance dataset was used for this project.

The dataset contains several hundred financial records representing individual expenses.

It includes key features such as Category, Amount, Date, and Savings Goal Status.

Date was further split into Year, Month, and Day for machine learning processing.

The dataset was refined and cleaned to suit the requirements of expense prediction and budget forecasting.

Important columns identified: Amount, Category, Date Components, Savings Goal.

PARSING AND CLEANING

Date Standardization & Extraction:

The raw date column was standardized into a proper datetime format.

Useful time-based features (**Year, Month, Day**) were extracted for forecasting.

The original date column was removed after extracting these components.

Removal of Non-Essential Attributes:

Attributes that did not contribute to prediction—such as **user_id**—were removed.

This reduced noise and simplified the feature space.

Missing Value Handling:

The dataset was scanned for incomplete entries.

Rows with missing values were removed to preserve data quality and prevent model instability.

Categorical Feature Preparation:

Categorical columns such as expense category and savings goal status were identified and prepared for encoding.

Encoding was performed using:

- CatBoost's built-in categorical handling

- LightGBM's category dtype

- OneHotEncoder for XGBoost

This ensured consistent treatment of categorical variables across all models.

Numerical Feature Consistency:

Numerical columns such as amount, year, month, and day were validated for correctness.

Any inconsistent or non-numeric values were corrected or removed to ensure stable model training.

This step helps maintain model reliability during forecasting and classification.

Final Clean Structured Dataset:

After preprocessing, the dataset became fully structured, free of missing values, and suitable for ML tasks.

The cleaned dataset enabled accurate training of CatBoost, LightGBM, XGBoost, and the stacked ensemble model.

FEATURE PREPARATION

PREPARING CATEGORICAL COLUMNS FOR MODEL COMPATIBILITY:

The categorical column was prepared in formats compatible with each ML model. (expense category)

Since CatBoost, XGBoost, and LightGBM handle categories differently, the column was transformed using:

- CatBoost's native categorical handling

- LightGBM's category dtype

- OneHotEncoding for XGBoost

This ensured smooth, error-free training across all models.

ALIGNING NUMERICAL FIELDS FOR STABLE LEARNING:

- Numerical fields such as amount, year, month, and day were validated for consistency and correctness.
- Only clean, valid numeric entries were retained to ensure stable model learning.
- This prevents the model from being influenced by irregular or noisy numerical values

CREATING A CLEAN, MINIMAL FEATURE SET:

The dataset was refined to retain only the features essential for prediction—category, amount, and date-derived fields (year, month, day).

Removing unused attributes ensured a focused feature set with reduced noise, improving model performance.

SYNCHRONIZING FEATURES ACROSS ALL MODELS:

Since CatBoost, XGBoost, and LightGBM handle inputs differently, the features were organized in a way that each model received a consistent and compatible representation of the same information.

CatBoost used native categorical handling, LightGBM used category dtype and XGBoost used OneHotEncoded features

This alignment supported fair comparison and enabled the stacked ensemble design.

Ensuring Model-Ready Feature Inputs:

Before training, all features were checked to confirm they were complete, correctly formatted, and free from inconsistencies.

This validation ensured the dataset was fully prepared for stable, accurate, and reliable model training.

TRAIN TEST SPLIT DATA

DIVIDING THE DATA FOR FAIR EVALUATION:

To ensure objective evaluation, the dataset was divided into separate training and testing portions.

This prevents the model from simply memorizing the data and helps measure how well it performs on unseen financial records.

The separation ensures reliable generalization during real-world usage.

TRAINING THE MODEL WITH 80% OF THE DATA:

80% of the cleaned dataset was used for training the machine learning models.

This portion allowed the models to learn meaningful patterns in expense categories, amounts, and date-based financial trends.

The remaining 20% was reserved for testing to evaluate prediction accuracy and overall model performance.

TESTING THE MODEL WITH 20% OF THE DATA:

The remaining 20% of the dataset was used as the test set.

This portion was never shown to the model during training, ensuring that the evaluation metrics reflect real world performance

The test set helps determine the accuracy and reliability of the model when encountering unseen data.

ENSURING CONSISTENT SPLITTING:

A random fixed state was used during the train-test split to maintain reproducibility.

This ensures that every run of the model uses the exact same division, making comparisons and debugging consistent throughout development.

Balanced Data Usage:

With this split, the model received enough data to learn from while keeping a fair portion aside as an unbiased benchmark.

DATA PREPARED FOR MODEL

After completing all stages of parsing, cleaning, and preparation, the dataset reached a fully structured and model-ready state. The categorical feature was formatted according to the specific requirements of CatBoost, XGBoost, and LightGBM, while numerical fields were validated for correctness and consistency. Irrelevant attributes were removed, and only meaningful predictors were retained to keep the feature space clean and focused.

The train–test split further ensured that the model learned from reliable data and was evaluated on unseen examples. With every feature checked, formatted, and properly structured for each ML framework, the dataset was now completely prepared for accurate and stable machine-learning training.

Model Selection & Overview:

The machine learning component of this project is built using three advanced gradient-boosting algorithms: CatBoost, XGBoost, and LightGBM. These models were selected because of their strong performance on tabular datasets, their ability to handle mixed numerical and categorical features, and their proven reliability in financial prediction tasks. CatBoost offers native support for categorical variables, reducing the need for heavy preprocessing and allowing the model to learn category interactions effectively. XGBoost, known for its robustness and precision, was included to capture deeper non-linear relationships by using OneHotEncoded features. LightGBM contributes high-speed training and efficiency through its leaf-wise growth strategy and category-aware data handling. Together, these models provide complementary strengths and form the basis for a powerful ensemble system. The objective behind selecting these three algorithms is to leverage their diverse learning behaviors to achieve more accurate, stable, and generalizable predictions for financial forecasting.

Individual Model Training:

Each machine learning model in this project was trained separately to understand spending patterns and contribute unique predictive strengths to the overall system. The CatBoost model was trained using the original categorical and numerical features with minimal preprocessing, allowing it to automatically handle category encodings and capture complex interactions within the data. XGBoost required a dedicated preprocessing pipeline in which categorical values were transformed using OneHotEncoding, ensuring that the model received fully numerical inputs suitable for gradient boosting. LightGBM was trained using the same core features but relied on converting categorical columns into the category dtype, enabling faster training and efficient tree construction. All three models were trained on 80% of the dataset, with consistent parameters such as controlled learning rates, optimized depth, and a fixed random seed to ensure stable and reproducible performance. Through this individual training process, each model learned different aspects of user spending behavior, laying the foundation for a more accurate and reliable ensemble approach.

Stacked Ensemble Architecture :

After training the individual models, their strengths were combined using a stacked ensemble approach to achieve higher prediction accuracy and better generalization. In this architecture, the first layer consists of the three base models—CatBoost, XGBoost, and LightGBM—which independently generate predictions using the same input features. Instead of relying on any single model, these outputs are then used as new input features for a second-level meta-model. CatBoost was selected as the meta-learner because of its ability to handle mixed input values and learn complex patterns from the base model predictions. This creates a multi-stage learning system where the first layer captures diverse perspectives on the data, and the second layer refines and integrates these insights into a more stable and accurate final prediction. The stacked ensemble effectively reduces the weaknesses of individual models, minimizes overfitting, and provides a more reliable forecasting mechanism, making it well-suited for financial prediction scenarios where precision and stability are crucial.

Model Evaluation and Comparison:

Once the individual models and the stacked ensemble were trained, their performance was evaluated using the 20% test split to determine how well they generalized to unseen data. Each model produced predictions on the test set, and these outputs were compared against the actual values to measure accuracy and overall reliability. CatBoost demonstrated strong performance due to its effective handling of categorical features, while XGBoost and LightGBM also produced competitive results through their optimized tree-based learning mechanisms. However, the stacked ensemble consistently achieved the highest accuracy because it combined the strengths of all three base models and reduced the likelihood of model-specific errors. The evaluation confirmed that the ensemble approach not only improved predictive stability but also delivered more consistent forecasts across different expense categories and time-based patterns. This comparative analysis validated the choice of using multiple models and reinforced the ensemble as the final and most reliable predictor for the system.

System Architecture:

The deployment of BudgetWise AI is structured around a lightweight yet efficient architecture that integrates the machine learning model with a fully interactive web application. At the core of the system is a Streamlit-based front-end interface, which handles user interaction, input processing, and visualization. This front-end communicates seamlessly with a backend powered by SQLite, a reliable and easy-to-manage local database used to store user information, expense records, budgets, and recurring transactions. The trained machine learning models, saved using joblib, are loaded directly into the Streamlit environment, enabling real-time predictions without requiring an external server. This integrated arrangement ensures fast response times and smooth operation, even on resource-limited systems. The architecture also includes secure authentication using SHA-256 hashing to protect user accounts, while the Gemini API is incorporated to offer AI-driven financial advice. Overall, the system architecture is designed to be modular, scalable, and simple to deploy, making it suitable for both personal use and small-scale production environments.

Web Application Features:

The deployed application provides a comprehensive suite of features designed to simplify financial management and enhance user experience. The system offers secure user authentication, allowing individuals to create accounts and access personalized financial data using encrypted credentials. Once logged in, users can add, edit, and delete expenses while optionally attaching receipt images for better record-keeping. The application also enables users to set monthly budgets for different categories and track their spending against these limits through intuitive visualizations. A recurring expense module automates entries for predictable costs such as rent, subscriptions, or utility payments, ensuring that all financial activity remains up-to-date without manual intervention. The integrated AI-powered forecasting tool allows users to select a category and view future spending predictions through dynamic line graphs covering the next 7 to 90 days. Additionally, the Gemini-based financial advisor analyzes recent spending patterns to provide personalized recommendations aimed at improving financial health. Together, these features create an intelligent, user-friendly platform that supports effective budgeting, financial awareness, and informed decision-making.

Model Integration and Backend Logic:

The integration of the machine learning model into the application is designed to operate seamlessly within the Streamlit environment, ensuring real-time predictions and smooth user interaction. Once the ensemble model is trained, it is saved using joblib and loaded directly when the application starts, eliminating the need for external servers or complex APIs. When a user inputs new expense information—such as category or date—the backend automatically processes these values into the same feature structure used during training, ensuring consistency and compatibility. For forecasting, the system generates future dates, applies the model to predict upcoming expenses, and returns these results to the front end, where they are visualized as dynamic line graphs. The backend also handles database operations, including storing and retrieving expenses, updating budgets, managing recurring transactions, and ensuring data integrity. Through this unified logic, the application maintains a robust pipeline from user input to model inference and visualization, enabling efficient, accurate, and responsive financial insights.

Hosting and Deployment Process:

The final stage of the project involves deploying the application in a way that makes it accessible to users without requiring any local setup. BudgetWise AI is hosted on Streamlit Cloud, a platform that allows seamless deployment of Python-based interactive applications. The entire project, including the Streamlit interface, the trained machine learning model, and the SQLite database schema, is packaged within a structured repository. When uploaded to Streamlit Cloud, the platform automatically installs all required dependencies listed in the requirements file and initializes the application environment. Sensitive information, such as the Gemini API key, is securely stored in Streamlit's built-in secrets management system, preventing exposure in the public codebase. Once deployed, the application runs on a managed server, loading the ML model at startup and serving the interactive dashboard to users through a simple web link. This hosting approach provides reliability, ease of access, and the scalability needed to support multiple users while maintaining consistent performance and security.