**PROJECT REPORT ON**

**ANALYTICS FOR BUSINESS FORECASTING**

#### **Carried Out at**

**CENTRE FOR DEVELOPMENT OF ADVANCED COMPUTING ELECTRONIC CITY, BANGALORE.**

### **UNDER THE SUPERVISION OF**

**MR. LAKSHAY**

**C-DAC Bangalore**

**Submitted By**

### **PALAK TOLWANI (250250125038)**

### **SHEETAL PATIL (250250125042)**

**ROHIT GUPTA (250250125051)**

**SAGAR KUMAR (250250125052)**

**PG DIPLOMA IN BIG DATA ANALYTICS C-DAC, BANGALORE**

# **Table of Contents**

|  |  |
| --- | --- |
| 1.0. **Project Title** | 1 |
| 2.0. **Introduction** | 1 |
| 3.0. **Objectives** | 2 |
| 4.0. **Overall Description**  4.1. Project Overview  4.2. Key Features  4.3. Benefits of the System | 3 |
| 5.0. **System Architecture**  5.1. Component Overview  5.2. System Flow Diagram | 4 |
| 6.0. **External Interface Requirements**  6.1. User Interfaces (Streamlit Dashboard)   6.2. Hardware Interfaces (User System, CPU/GPU)   6.3. Software Interfaces (Python Libraries and Models)   6.4. Communication Interfaces (Session State & Caching) | 7 |
| 7.0. **System Features and Core Components**  7.1. Dataset Upload and Cleaning   7.2. Exploratory Data Analysis (EDA) and Visualization   7.3. Correlation Analysis and Feature Selection   7.4. Time Series Forecasting     7.4.1. Prophet Model     7.4.2. ARIMA Model     7.4.3. XGBoost Regressor     7.4.4. Linear Regression     7.4.5. Random Forest   7.5. Lag Feature Engineering   7.6. Forecast Visualization and Evaluation Metrics   7.7. Profit Forecasting   7.8. UI Design and User Flow | 9 |
| 8.0. **System Architecture and Workflows**  8.1. Data Flow Diagram  8.2. Workflow Example | 12 |
| 9.0. **Testing and Validation**  9.1. Functional Testing   9.2. Forecast Accuracy Evaluation (MAE, RMSE, MAPE)   9.3. Error Handling | 14 |
| 10.0. **Limitations and Future Work**  10.1. Known Limitations  10.2. Planned Enhancements | 16 |
| 11.0. **Developer Documentation**  11.1. Key Functions and Modules  11.2. Customization Options | 18 |
| 12.0. **Progress Report / Timeline** | 21 |
| 13.0. **Conclusion** | 22 |
| 14.0. **References** | 23 |
| 12.0. **Non-Functional Requirements** | 24 |
| 13.0. **Acceptance Criteria** | 25 |
| 14.0. **Deliverables** | 26 |

**1. Project Title**

**Analytics for Business Forecasting**

This project presents a comprehensive, user-friendly web application for time series forecasting using Streamlit. It allows businesses to forecast future sales, analyze the impact of features like promotions and inventory, and visualize profits. It supports multiple models including Prophet, ARIMA, XGBoost, Linear Regression, and Random Forest.

**2. Introduction**

In today's rapidly evolving retail landscape, businesses increasingly rely on data-driven insights to make informed decisions. Forecasting plays a crucial role in anticipating future demand, optimizing inventory, planning promotions, and maximizing profits. However, building accurate and interpretable forecasting models often requires technical expertise and complex tooling.

The project **“Analytics for Business Forecasting”** aims to bridge this gap by providing an interactive, intuitive web-based dashboard for forecasting sales and analyzing business performance. Built using Python and Streamlit, the application empowers users to upload their sales data, perform exploratory analysis, apply different forecasting models, and visualize predictions — all without writing a single line of code.

It supports a wide range of models including:

* **Prophet** for trend-seasonality based forecasting,
* **ARIMA** for traditional statistical modeling,
* **XGBoost, Linear Regression, and Random Forest** for machine learning-based regression.

Additionally, the system offers features like:

* Auto-handling of missing values,
* Lag feature creation,
* Custom visualizations (histograms, box plots, scatter plots),
* Correlation heatmaps,
* Profit forecasting using a markup percentage.

The goal is to provide business users with a powerful yet simple tool that helps them forecast future trends and make proactive decisions based on data.

**3. Objectives**

The primary objective of the **Analytics for Business Forecasting** project is to develop a user-friendly, interactive forecasting platform that allows businesses to make accurate, data-driven decisions regarding sales, inventory, and profitability.

**Primary Objectives**

* Enable Sales Forecasting  
  Allow users to forecast future sales using multiple models including Prophet, ARIMA, XGBoost, Linear Regression, and Random Forest.
* Interactive Dashboard  
  Build a Streamlit-based web application that supports real-time data interaction, visualization, and model selection.
* Data Upload and Preprocessing  
  Facilitate easy upload of CSV datasets with automatic handling of missing values and data cleaning.
* Model Comparison and Accuracy Metrics  
  Provide visual comparison of forecasts and evaluate performance using MAE, RMSE, and MAPE.
* Profit Forecasting  
  Calculate projected profit based on forecasted sales and customizable markup percentages.

**Secondary Objectives**

* Lag Feature Engineering  
  Allow users to configure and apply lag-based features for time series regression models.
* Feature Correlation Analysis  
  Provide heatmaps to help identify important predictors and guide feature selection.
* Custom Data Visualizations  
  Enable users to create histograms, box plots, scatter plots, and categorical bar charts for exploratory data analysis.
* Downloadable Results  
  Allow users to download both forecasted sales and profit data in CSV format.
* Support for Categorical and Numerical Features  
  Allow one-hot encoding for categorical variables and incorporation of numerical features into forecasting pipelines.

**4. Overall Description**

The **Analytics for Business Forecasting** project provides a complete, modular solution for forecasting sales and analyzing business performance trends through an interactive web-based dashboard. It is designed for users with little to no programming experience and offers a clean, guided interface powered by **Streamlit**. Users can upload their datasets, visualize trends, select forecasting models, and view or download the results.

The application incorporates both traditional and machine learning forecasting techniques, including:

* **Prophet** for trend and seasonality modeling,
* **ARIMA** for statistical time series analysis,
* **XGBoost, Linear Regression, and Random Forest** for regression-based forecasting.

It also supports **profit projection** based on markup percentage and evaluates model performance using industry-standard metrics.

**4.1 Project Overview**

The system operates entirely through a browser interface and supports the following end-to-end pipeline:

1. **Upload CSV Data**: Users upload a dataset containing historical sales and associated features.
2. **Preprocessing**: The application handles missing values automatically.
3. **EDA and Feature Engineering**: Correlation heatmaps, custom visualizations, and lag feature selection are provided.
4. **Model Selection**: Users choose from 5 forecasting models, each with its own configurable options.
5. **Forecast Generation**: Future sales values are predicted and plotted alongside historical data.
6. **Profit Forecasting**: Users can apply a markup percentage to estimate potential profit based on sales forecasts.
7. **Output Export**: Forecasted data can be downloaded for reporting or analysis.

This modular design allows for flexible experimentation and fast iteration for business analysts.

**4.2 Key Features**

* Streamlit-based interactive UI
* CSV file upload and automatic data cleaning
* Date and target (sales) column selection
* Prophet, ARIMA, XGBoost, Linear Regression, Random Forest model support
* Lag feature engineering
* One-hot encoding of categorical features
* Custom visualizations: histogram, scatter, box plots
* Correlation matrix generation
* Forecasting metrics: MAE, RMSE, MAPE
* Profit calculation based on user-defined markup
* Forecast and profit CSV download

**4.3 Benefits of the System**

* No-Code Interface: Business users can run advanced forecasting without programming knowledge.
* Flexible Model Comparison: Users can test and compare multiple models in a single session.
* Visual Guidance: Integrated charts and metrics guide decision-making.
* Profit Planning: Helps estimate expected revenue based on forecasted sales.
* Scalable and Portable: Runs on local machines and can be extended for enterprise use.
* Offline Capable: Can operate entirely offline after setup, ensuring data privacy.

**5. System Architecture**

The **Analytics for Business Forecasting** system is architected as a modular and interactive dashboard that supports end-to-end time series forecasting, model evaluation, and business insight generation. It integrates preprocessing, visualization, model training, and forecasting workflows in a streamlined user interface powered by **Streamlit**.

The architecture is composed of five major layers:

1. **Data Input and Cleaning Layer**
2. **Feature Engineering and Transformation Layer**
3. **Forecasting Engine (Model Layer)**
4. **Evaluation and Profit Computation Layer**
5. **User Interface (Streamlit Layer)**

Each layer is responsible for a set of tasks, with communication managed via in-memory operations and st.session\_state for persistent user interaction across steps.

**5.1 Component Overview**

* CSV Upload Module  
  Allows users to upload datasets in .csv format containing sales and other business variables.
* Preprocessing Engine  
  Handles missing value imputation, duplicate removal, and parsing of date columns. Automatically infers data types and cleans the dataset.
* Feature Engineering Module  
  Enables lag creation for time series models, and one-hot encodes selected categorical variables. Computes correlation matrices to aid feature selection.
* Modeling Layer
* Provides forecasting using:
* Prophet (additive model with seasonality/trend)
* ARIMA (statistical modeling)
* XGBoost, Linear Regression, Random Forest (machine learning regressors with recursive forecasting)

**Each model handles training, prediction, and future extrapolation differently.**

* Profit Estimation Module  
  Uses forecasted sales and user-defined markup percentage to compute and display projected profit.
* Metrics Engine  
  Calculates MAE, RMSE, and MAPE for model evaluation using recent actuals and forecast outputs.
* Visualization Engine  
  Generates interactive line charts, scatter plots, histograms, box plots, and bar charts for both raw and forecasted data.
* Download Engine  
  Enables users to export forecast and profit data as .csv files.
* Session Management (State)  
  All inputs, outputs, models, and selections are tracked using st.session\_state for continuity.

**5.2 System Flow Diagram**

The **Analytics for Business Forecasting** system follows a structured sequential workflow — from data upload to forecast generation and profit computation. The entire process is handled interactively through a Streamlit web interface. The key stages are as follows:

#### **1. CSV Upload and Data Cleaning**

● User uploads a .csv file containing historical sales data along with features such as date, promotion status, inventory, etc.

● The system checks for missing values and fills them automatically using mean (numeric) or mode (categorical).

● Duplicate rows are removed, and date columns are parsed.

● Cleaned data is stored in session state for further processing.

#### **2. Feature Selection and Preprocessing**

● User selects the date column, target column (e.g., sales), and any additional numeric or categorical features.

● Categorical features are one-hot encoded.

● Lag features are generated for regression models (user-configurable).

● Correlation heatmaps are displayed to assist with feature importance analysis.

**3. Forecast Generation and Visualization**

**●** The chosen model is trained on the historical dataset.

● Future values are forecasted for a user-selected time period (e.g., 30 days).

● Line charts display both actual historical sales and forecasted future sales.

● Metrics like MAE, RMSE, and MAPE are computed and displayed for performance evaluation.

#### **5. Profit Forecasting**

● User selects an inventory column (optional) and a promotion column (numeric or categorical).

● A **markup percentage slider** is provided to simulate profit margins.

● Profit is calculated as:Predicted Sales × Markup (%)

● Forecasted profit values are displayed in tabular form and can be downloaded.

**6. Result Output and Feedback**

● Forecasted sales and profit tables can be downloaded as .csv files.

● The application handles invalid inputs, missing values, and edge cases with appropriate alerts.

● Streamlit widgets and session state provide real-time interactivity without restarting the app.

**6. External Interface Requirements**

This section outlines the external interfaces used by the **Analytics for Business Forecasting** system. These include user interaction components, hardware dependencies, software libraries, and the flow of data between internal modules. The system is designed to run locally on a personal computer with Python installed, making it accessible, lightweight, and easy to deploy.

**6.1. User Interfaces (Streamlit Dashboard)**

* The application uses Streamlit to create an interactive, browser-based user interface.
* Users interact through:
  + Sidebar controls for file upload, feature selection, model choice, and markup input.
  + Main panel visualizations for displaying charts, heatmaps, and forecast results.
* No command-line interface is required — all operations are performed through buttons, dropdowns, sliders, and selection boxes.
* Forecast and profit data are available for download in .csv format with a single click.
* Real-time updates and session memory allow users to switch models or features without resetting the app.

**6.2. Hardware Interfaces (User System, CPU/GPU)**

* Designed to run on local desktop or laptop systems.
* Minimum requirements:
  + Processor: Intel i3 or higher
  + Memory: 4 GB RAM minimum (8 GB recommended)
  + Storage: ~500 MB for Python environment and dependencies
* GPU is not required, but optional acceleration (e.g., XGBoost with GPU) can be enabled if supported.
* Compatible with Windows, Linux, and macOS environments.

**6.3. Software Interfaces (Python Libraries and Models)**

The system relies on a number of Python packages for forecasting, visualization, and UI:

* Streamlit – UI and interactivity
* Pandas, NumPy – Data manipulation and handling
* Matplotlib, Seaborn – Visualization of time series and correlations
* Prophet – Trend and seasonality forecasting
* Statsmodels (ARIMA) – Statistical time series modeling
* XGBoost – Gradient boosting model
* Scikit-learn – Linear Regression, Random Forest, and error metrics
* Pickle – For model serialization if needed (currently unused in UI)

**6.4. Communication Interfaces (Session State & Caching)**

* Internal components communicate via Streamlit’s st.session\_state to store:
  + Cleaned datasets
  + Feature selections
  + Trained models and configurations
  + Forecast and profit results
* Data does not leave the local environment, ensuring privacy and offline capability.
* Session memory allows seamless switching between forecasting models and avoids redundant reprocessing.

**7. System Features and Core Components**

he **Analytics for Business Forecasting** system is a full-featured, interactive forecasting tool designed to enable business users to explore, analyze, and forecast sales data with minimal technical knowledge. It integrates data preprocessing, visualization, machine learning, and forecasting logic into a single, unified web application built using **Streamlit**.

Below is a breakdown of its key features and core functional components:

**7.1. Dataset Upload and Cleaning**

* Users can upload sales datasets in .csv format.
* Automatic handling of missing values:
  + Numeric columns: filled with column mean
  + Categorical columns: filled with mode or "Unknown"
* Duplicate records are removed to ensure data consistency.
* Date column is parsed and sorted chronologically for time series forecasting.

**7.2. Exploratory Data Analysis (EDA) and Visualization**

* Users can explore trends using various plot types:
  + Histogram with KDE option
  + Box Plot for outlier analysis
  + Scatter Plot with color-based grouping
  + Categorical Bar Charts for distribution insight
* Grouped bar charts show percentage comparisons between categorical fields.
* All plots are dynamically updated based on user selections.

**7.3. Correlation Analysis and Feature Selection**

* Generates a correlation heatmap to analyze relationships between numeric features and the target (e.g., sales).
* Helps users identify strongly correlated variables to improve model accuracy.
* Users can select numeric and categorical features for forecasting.
* Categorical features are automatically one-hot encoded.

**7.4. Time Series Forecasting**

Supports five forecasting algorithms:

**7.4.1. Prophet Model**

* Detects trend, seasonality, and holiday effects.
* Allows inclusion of additional regressors like promotions, inventory, etc.
* Generates forecasts with daily granularity for future periods.

**7.4.2. ARIMA Model**

* Performs autoregressive integrated moving average modeling.
* Ideal for stationary time series with consistent patterns.
* Includes error handling for short or noisy datasets.

**7.4.3. XGBoost Regressor**

* Ensemble machine learning model for high accuracy.
* Uses lag-based features and other predictors.
* Performs **recursive forecasting** day-by-day into the future.

**7.4.4. Linear Regression**

* A lightweight and interpretable forecasting model.
* Uses lag and additional predictors to forecast future values.

**7.4.5. Random Forest**

* A tree-based ensemble model suitable for non-linear relationships.
* Handles both numeric and categorical inputs efficiently.

**7.5. Lag Feature Engineering**

* User-defined number of lag days can be added as features (up to 10).
* Lag features are critical for XGBoost, Linear Regression, and Random Forest models.
* Lagging helps capture past sales trends and cyclic patterns.

**7.6. Forecast Visualization and Evaluation Metrics**

* Forecasts are plotted alongside historical sales for direct visual comparison.
* Displays:
  + Actual vs. Predicted Sales Line Chart
  + MAE – Mean Absolute Error
  + RMSE – Root Mean Square Error
  + MAPE – Mean Absolute Percentage Error
* Forecast outputs are downloadable as CSV files.

**7.7. Profit Forecasting**

* Users can select an inventory column and optionally a promotion column.
* A markup percentage slider is used to calculate profit:  
  Profit = Predicted Sales × Markup (%)
* Profit values are shown in a table and available for download.

**7.8. UI Design and User Flow**

* Built using **Streamlit** with a responsive layout and intuitive sidebar controls.
* Real-time interactivity with:
  + Forecast horizon selection
  + Model choice
  + Dynamic plots and metrics
* Session state stores user selections and results, allowing seamless navigation without reprocessing.
* Alerts and warnings guide users through valid inputs and catch common errors.

**8. System Architecture and Workflows**

The architecture of the **Analytics for Business Forecasting** system is modular, scalable, and optimized for ease of use. It allows business users to perform complex forecasting tasks through an interactive dashboard built with **Streamlit**, without requiring any coding or backend configuration.

This section outlines both the **logical system architecture** and the **workflow** a typical user follows when using the application.

* **8.1 Data Flow Diagram**  
  ● The system follows a logical and sequential flow from dataset ingestion to profit prediction, ensuring modularity, performance, and ease of user interaction.
* A high-level data flow can be visualized through the following stages:

mathematica

CSV Upload → Data Cleaning → Feature Selection → Feature Engineering →

Model Selection → Forecast Generation → Metric Evaluation → Profit Forecasting → Output Download

* Data preprocessing is performed once per session. Forecasting and profit projection are repeated based on user interactions.
* The system maintains state using st.session\_state, but the forecasting flow remains **stateless and modular** between components.
* **Figure 8.1 (to be added)** may represent this data flow using standard DFD notations (Level 1 or Level 2), illustrating components such as:
  + Data Input Module
  + Preprocessing Engine
  + Feature Engineering Module
  + Forecasting Models
  + Evaluation Engine
  + Visualization & Download Manager

**8.2 Workflow Example**  
● **8.2 Workflow Example**

* The following step-by-step workflow illustrates how a typical user interacts with the system from data upload to insight generation:

**1. Dataset Upload**

* The user uploads a .csv file containing historical sales records.
* The system parses the data, identifies numeric and categorical columns, and handles missing values automatically.
* Date columns are converted to datetime objects and sorted for proper sequencing.

**2. Data Exploration and Feature Selection**

* Users explore the dataset visually using histograms, box plots, and scatter plots.
* A correlation heatmap is generated to help identify useful predictors.
* The user selects the **date** column, **target** (e.g., sales), and any additional categorical/numeric features.

**3. Lag Feature Engineering (Optional)**

* For regression models, the user selects how many lag days to include.
* The system creates lagged versions of the target variable to use as predictors.
* Categorical features are one-hot encoded automatically.

**4. Model Selection and Forecasting**

* The user chooses one of five models: Prophet, ARIMA, XGBoost, Linear Regression, or Random Forest.
* The selected model is trained on the available data.
* Forecasts for a user-defined number of days are generated.

**5. Evaluation and Visualization**

* Forecasts are compared against recent actual data using MAE, RMSE, and MAPE.
* Visual charts show actual vs. predicted sales, enabling intuitive analysis.
* Each model run is cached using Streamlit to prevent unnecessary recomputation.

**6. Profit Forecasting**

* The user applies a **markup percentage** using a slider.
* Predicted profit is calculated based on the forecasted sales.
* Profit data is displayed in tabular format and is available for download.

**7. Result Display and Output**

* Forecasts and profit predictions are shown alongside confidence metrics.
* Results can be downloaded in .csv format for business reporting or planning.
* The system remains active in memory for new uploads, experiments, or repeated testing.

**9. Testing and Validation**

Thorough testing and validation are critical to ensure the correctness, robustness, and reliability of the **Analytics for Business Forecasting** system. The application has been tested across multiple stages, including data processing, model training, forecasting accuracy, and user interface interaction. Both functional and performance aspects were validated to guarantee a seamless user experience.

**9.1 Functional Testing**  
● The following modules and functionalities were tested independently and in combination:

* **File Upload**: Tested with valid and invalid .csv files (e.g., missing headers, null entries, unsupported formats).
* **Date and Target Column Selection**: Validated proper dropdown population and error messaging for missing columns.
* **Exploratory Data Analysis**: Verified correctness of visual outputs (histograms, scatter plots, box plots, heatmaps).
* **Feature Selection**: Ensured only numerical and categorical columns are selectable with accurate encoding.
* **Model Selection and Execution**: All five forecasting models (Prophet, ARIMA, XGBoost, Linear Regression, Random Forest) tested across multiple datasets.
* **Forecast Output**: Ensured predicted values match future time windows and no leakage occurs.
* **Profit Calculation**: Verified profit computation based on markup input and accurate formatting in output table.
* **Download Functionality**: Tested successful CSV export for forecast and profit outputs.

**9.2 Accuracy and Confidence Validation**  
● Model accuracy was assessed using three widely accepted metrics:

* **MAE (Mean Absolute Error)** – Indicates average magnitude of errors in prediction.
* **RMSE (Root Mean Square Error)** – Penalizes larger errors more than MAE; useful for retail fluctuations.
* **MAPE (Mean Absolute Percentage Error)** – Measures error as a percentage of actual values; easily interpretable.

Each model’s output was compared against actual historical sales over the last portion of the dataset (holdout testing). Accuracy scores were displayed for each model, allowing users to compare performance visually and numerically.

**9.3 Error Handling**  
● Robust exception handling was integrated and tested in the following scenarios:

* **Missing Values in Critical Columns**: Warns the user and fills values with mean/mode as fallback.
* **Short Time Series (<30 rows)**: Notifies the user that models like ARIMA/Prophet need more data.
* **Non-Date Columns Misclassified**: Prevents selection of string/text columns as date/time.
* **Model Failures**: Captures exceptions from ARIMA model fitting and prevents crashes.
* **Forecast Range Too Large**: Limits user input to a reasonable number of days to avoid memory overload.
* **Unsupported File Format**: Blocks and notifies on files that are not .csv.

**10. Limitations and Future Work**

While the **Analytics for Business Forecasting** system provides a powerful and user-friendly platform for sales forecasting, it also has a few limitations that can be addressed in future versions. These limitations are primarily due to the simplified assumptions made for general usability and the constraints of browser-based execution.

**10.1 Known Limitations**  
 **Forecasting Horizon Dependency**

* The accuracy of long-term forecasts may degrade, especially for highly volatile or seasonal datasets.
* Recursive models like XGBoost and Random Forest accumulate prediction error over time.

**Limited Hyperparameter Tuning**

* The current implementation uses default or static model parameters for ease of use.
* Users cannot fine-tune hyperparameters (e.g., ARIMA order, XGBoost depth) within the interface.

**No Cross-Validation or Automated Backtesting**

* The app evaluates model accuracy on a static holdout set.
* Cross-validation techniques (e.g., rolling forecast origin) are not yet implemented.

**Categorical Feature Limitations**

* Categorical features are one-hot encoded, which may not scale well with high-cardinality features.
* There is no option to group or reduce dimensionality of categorical data.

**Model Training on Single Session**

* All computations are session-based and run in memory.
* No support for persistent model saving, parallel training, or batch processing.

**UI Responsiveness on Large Datasets**

* Performance may drop when visualizing or processing very large datasets (>50K rows).
* Model training for large time windows can take longer in resource-constrained systems.

**10.2 Planned Enhancements**  
 **Hyperparameter Tuning Interface**

* Introduce sliders and advanced tabs for users to customize model parameters such as ARIMA (p,d,q), Prophet changepoints, and XGBoost estimators.

**Model Saving and Loading**

* Add functionality to save trained models and reload them for future predictions without retraining.

**Cross-Validation and Backtesting**

* Implement time-based cross-validation and rolling origin backtesting to improve model robustness.

**Automated Model Selection**

* Add scoring mechanism to automatically recommend the best model based on past performance and data characteristics.

**Deployment to Cloud or API Access**

* Extend the app to support deployment on platforms like Heroku or Streamlit Cloud with API access for automated forecasting.

**Multi-Metric Optimization**

* Provide an option for users to optimize models based on a chosen metric (e.g., minimize RMSE, maximize profit).

**Seasonal Decomposition and Advanced Feature Extraction**

* Add functionality for STL decomposition and lag autocorrelation plots to better understand time series components.

**Mobile-Friendly UI**

* Optimize the layout and responsiveness of the Streamlit dashboard for mobile devices.

**11. Developer Documentation**

The **Analytics for Business Forecasting** system is developed entirely in **Python** using the **Streamlit** framework for UI. It is designed with modularity and extensibility in mind, allowing future developers to easily maintain or expand the system. Below is technical documentation covering the core functions, structure, and customization options for the developers.

**11.1 Key Functions and Modules**  
**● main()**

* Entry point for the Streamlit application.
* Handles layout rendering, sidebar navigation, session state initialization, and all user interactions.
* Coordinates file upload, data preprocessing, feature selection, model selection, forecasting, and profit calculation.

**● create\_lag\_features(data, target\_col, n\_lags)**

* Generates lagged versions of the target column (e.g., sales) for supervised learning.
* Iteratively creates columns like sales\_lag\_1, sales\_lag\_2, etc., used in ML models.
* Returns a modified DataFrame with lagged features.

**● train\_and\_forecast\_model(model\_name, df, target\_col, features, forecast\_days)**

* Handles training and prediction logic for each selected model:
  + Prophet
  + ARIMA
  + XGBoost
  + Random Forest
  + Linear Regression
* Selects the appropriate model pipeline based on model\_name.
* Returns predicted values and evaluation metrics (MAE, RMSE, MAPE)

**● calculate\_profit(predicted\_sales, markup)**

* Computes profit projection based on forecasted sales and user-defined markup.
* Formula: Profit = Predicted Sales × (Markup % / 100)
* Returns a list of profit values for the forecast horizon.

**● evaluate\_model(y\_true, y\_pred)**

* Calculates evaluation metrics to assess forecast accuracy.
* Metrics used:
  + **MAE** (Mean Absolute Error)
  + **RMSE** (Root Mean Square Error)
  + **MAPE** (Mean Absolute Percentage Error)
* Results displayed in a user-friendly format within the app.

**● plot\_forecast(df, forecast\_df)**

* Visualizes historical and forecasted sales using Matplotlib or Streamlit charts.
* Enhances interpretability with color-coded lines and legends.

**● render\_custom\_visualizations(df)**

* Generates histograms, scatter plots, box plots, bar charts, and correlation heatmaps.
* Helps users explore relationships between features and identify trends.

**● handle\_file\_upload()**

* Accepts and parses user-uploaded .csv files.
* Validates column types and handles missing values.
* Stores the cleaned dataset into st.session\_state.

**11.2 Customization Options**

**● Adjusting Forecast Horizon**

* Developers can change the default range of days available for forecasting.
* Modify the slider limits or allow user-defined inputs via st.number\_input.

**● Adding New Forecasting Models**

* To integrate new models (e.g., LSTM, SARIMA):
  + Create a new branch in train\_and\_forecast\_model().
  + Define training logic and required preprocessing steps.
* Ensure compatibility with feature selection and lag inputs.

**● Improving Feature Engineering**

* Introduce additional time-based features like:
  + Day of week, month, seasonality flags
  + Rolling means, exponentially weighted averages
* Enhance model learning for cyclic or seasonal datasets.

**● UI Enhancements**

* Streamlit allows embedding:
  + Tabs for separating steps (Upload, Model, Forecast)
  + Sidebar sliders for hyperparameter tuning (e.g., ARIMA p, d, q)
  + Expanders for detailed error logs or documentation
* Developers can add interactive markdown, emojis, and Lottie animations for a better UX.

**● Enabling GPU Support (for XGBoost)**

* XGBoost supports GPU acceleration.
* Pass tree\_method="gpu\_hist" and ensure CUDA environment is set up.
* Improves training time for large datasets.

**● Logging and Debugging**

* Use Python’s logging module to capture:
  + Runtime errors
  + Data inconsistencies
  + Model performance logs
* Developers can toggle verbosity for debugging or production environments.

**● Session Persistence**

* Streamlit st.session\_state manages:
  + Uploaded files
  + Cleaned data
  + Model results and forecasts
* Persistent state allows seamless switching between model trials without reloading data.

**12. Progress Report / Timeline**

The **Analytics for Business Forecasting** project was executed using a modular and iterative development strategy. Each stage of the project—from ideation to deployment—was carefully planned, implemented, and tested before moving on to the next. Continuous collaboration and testing ensured a stable, functional, and user-friendly system by the end of the development cycle.

The timeline below summarizes the major milestones and their completion status:

| **Week** | **Task / Milestone** | **Status** |
| --- | --- | --- |
| Week 1 | |  | | --- | | Project idea finalization and definition of problem statement |  |  | | --- | |  | | Completed |
| Week 2 | |  | | --- | | Dataset collection and exploration (sample sales data in CSV format) |  |  | | --- | |  | | Completed |
| Week 3 | |  | | --- | | Environment setup (Python, Streamlit, Prophet, ARIMA, XGBoost, etc.) |  |  | | --- | |  | | Completed |
| Week 4 | |  | | --- | | Implementation of CSV upload and data cleaning module |  |  | | --- | |  | | Completed |
| Week 5 | |  | | --- | | Feature selection UI, EDA visualizations, and correlation heatmap |  |  | | --- | |  | | Completed |
| Week 6 | |  | | --- | | Integration of forecasting models: Prophet, ARIMA |  |  | | --- | |  | | Completed |
| Week 7 | |  | | --- | | Development of lag feature creation and regression model support |  |  | | --- | |  | | Completed |
| Week 8 | |  | | --- | | Forecast generation and visual display of results |  |  | | --- | |  | | Completed |
| Week 9 | |  | | --- | | Error metric computation and model comparison logic |  |  | | --- | |  | | Completed |
| Week 10 | |  | | --- | | Profit forecasting and markup calculation feature |  |  | | --- | |  | | Completed |
| Week 11 | |  | | --- | | Final UI enhancements, session state logic, and bug fixes |  |  | | --- | |  | | Completed |
| Week 12 | |  | | --- | | Documentation, report writing, and presentation slide creation |  |  | | --- | |  | | Completed |

The project was successfully completed over a 12-week development cycle. Agile methodology was followed, with frequent reviews and feedback sessions to ensure the system met both technical and user-centric requirements.

**13. Conclusion**

The **Analytics for Business Forecasting** project successfully achieved its objective of building an interactive, flexible, and accurate forecasting platform tailored for small businesses and analysts. By integrating multiple forecasting models (Prophet, ARIMA, XGBoost, Linear Regression, and Random Forest) with powerful visual analytics and profit estimation tools, the application provides end-users with actionable insights into future sales performance.

Through the use of Streamlit, the platform delivers a user-friendly experience with intuitive controls for uploading data, selecting features, choosing models, and interpreting results. The modular design ensures the system is scalable and extensible for future enhancements such as automated model tuning, advanced seasonal decomposition, or cloud deployment.

Key accomplishments of this project include:

* A complete forecasting pipeline from data upload to future prediction and profit calculation.
* Visualizations for EDA, model accuracy evaluation, and forecast presentation.
* A session-aware, real-time dashboard requiring no coding from the user’s side.
* The ability to run and compare multiple forecasting approaches under one unified interface.

This tool has real-world applications in inventory management, sales planning, budget forecasting, and business intelligence. Its simplicity, performance, and adaptability make it suitable for local businesses looking to gain a competitive edge through data-driven strategies.

The project stands as a strong foundation for continued innovation in AI-driven business forecasting systems.

**14. References**

* Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and Practice* (2nd ed).  
  <https://otexts.com/fpp2/>
* Prophet Model Documentation – Meta (Facebook)  
  <https://facebook.github.io/prophet/>
* ARIMA Model – Statsmodels Library  
  <https://www.statsmodels.org/stable/generated/statsmodels.tsa.arima.model.ARIMA.html>
* XGBoost Documentation  
  <https://xgboost.readthedocs.io/>
* Scikit-learn Documentation (Metrics & Regression Models)  
  <https://scikit-learn.org/stable/documentation.html>
* Streamlit Documentation  
  <https://docs.streamlit.io/>
* Python Official Documentation  
  <https://docs.python.org/3/>

**15. Non-Functional Requirements**

The **Analytics for Business Forecasting** system is designed not only to deliver accurate forecasts but also to ensure a seamless and efficient experience for users across varying environments. The following non-functional requirements outline the quality attributes that the system adheres to:

**15.1 Performance Requirements**

* The system should generate forecasts within **3–5 seconds** for small to medium datasets (under 10,000 rows).
* All visualizations and metrics should update dynamically without page reloads.
* Cached components (e.g., models and embeddings) should reduce redundant processing.

**15.2 Usability Requirements**

* The user interface should be intuitive, requiring **no programming knowledge**.
* Clear labeling, tooltips, and dynamic warnings guide the user through each step.
* Results such as forecast data and profit calculations must be easily interpretable.

**15.3 Scalability**

* The system should support datasets up to **50,000 rows** for real-time processing.
* Architecture allows for future integration of more models and visual modules without redesigning the core logic.

**15.4 Reliability and Fault Tolerance**

* Graceful error handling is implemented for:
  + Missing values
  + Invalid file types
  + Short datasets unsuitable for time series
* The application provides meaningful warnings and fallback strategies (e.g., data filling).

**15.5 Maintainability**

* Code is modular and documented with clearly named functions and logical blocks.
* Easy to update forecasting models or add new preprocessing steps.
* Variables and configurations can be adjusted without altering core logic.

**15.6 Portability**

* The system can run on any platform with Python and Streamlit installed.
* Easily deployable on cloud platforms like Streamlit Cloud, Heroku, or local servers.
* No dependencies on proprietary or OS-specific features.

**15.7 Security**

* Application runs locally or in controlled environments (not exposed to public internet by default).
* No external data transmission; uploaded data remains in-session.
* No sensitive credentials or third-party APIs are required.

**16. Acceptance Criteria**

The **Analytics for Business Forecasting** project will be considered successfully completed upon fulfilling the following measurable acceptance criteria. These criteria are derived from the project’s functional and non-functional goals and reflect its usability, performance, and correctness.

**1. Data Upload and Preprocessing**

* Users can upload a .csv file via the interface without errors.
* The system correctly reads the file and displays column headers for selection.
* Missing values are handled gracefully with default imputation (mean/mode).

**2. Feature Selection and Visualization**

* Users can select:
  + A **date column**
  + A **target column**
  + Optional **numerical** and **categorical** features
* Users can view visualizations (histogram, box plot, scatter plot, heatmap) based on the uploaded data.

**3. Forecasting Functionality**

* Users can select from five forecasting models: **Prophet, ARIMA, XGBoost, Linear Regression, Random Forest**.
* The system generates forecasts for the selected number of future days.
* Forecasted results are displayed in a clear, time-series chart format.

**4. Model Evaluation**

* Forecasts are evaluated and displayed with accuracy metrics:
  + MAE
  + RMSE
  + MAPE
* Metrics update dynamically based on the selected model and input features.

**5. Profit Forecasting**

* Users can enter a **markup percentage** using a slider.
* Forecasted profit is calculated and displayed in tabular format.
* Users can download both forecast and profit data in .csv format.

**6. UI and Interactivity**

* Application loads within 10 seconds on local or cloud deployment.
* User interactions (model selection, sliders, toggles) trigger real-time responses without reloading the page.
* Error messages and success notifications are clearly displayed.

**7. Robustness and Error Handling**

* Application gracefully handles:
  + Incomplete datasets
  + Incorrect file formats
  + Insufficient data length for time series
* Users are notified with clear, actionable warnings.

**8. Portability**

* The app runs successfully on Windows, Linux, and macOS systems with Python and required packages.
* Deployable on platforms like **Streamlit Cloud** or **Heroku** without major modifications.

**17. Deliverables**

The **Analytics for Business Forecasting** project includes a set of well-defined deliverables that demonstrate the system's capabilities, functionality, and readiness for deployment. These deliverables include documentation, source code, working application files, and supporting materials required for evaluation and presentation.

**1. Final Project Report**

* A comprehensive documentation of the project including:
  + Problem statement
  + System architecture
  + Functional and non-functional requirements
  + Implementation details
  + Testing, validation, and results
  + Conclusion and future scop

**2. Source Code**

* A complete and well-commented Python script (app.py) that includes:
  + Data upload and preprocessing modules
  + Forecasting model integration (Prophet, ARIMA, XGBoost, Random Forest, Linear Regression)
  + Feature engineering (lag features)
  + Profit forecasting logic
  + Visualization and download functionalities

**3. Sample Dataset**

* A sample .csv dataset (e.g., sales records) used for testing the application.
* Ensures reviewers can test the forecasting pipeline end-to-end.

**4. Forecast Output Samples**

* Forecasted data and profit projections in .csv format.
* Includes output from various models to demonstrate differences in accuracy.

**5. Presentation Slides**

* A concise PowerPoint presentation covering:
  + Project overview
  + Objective and motivation
  + Model comparisons
  + Key results and visualizations
  + Demo screenshots and architecture diagrams

**6. Deployment Instructions**

* Step-by-step instructions to set up the environment:
  + Required Python version and packages (requirements.txt)
  + How to run the Streamlit app locally
  + Optional: Deployment on Streamlit Cloud or Heroku