

Bargain Bot



Project Supervisor:

Dr. Samyan Qayyum Wahla

Project Members

Gul-e-Zahra	2022-CS-75
Amna Nadeem	2022-CS-96

Department of Computer Science
University of Engineering and Technology
Lahore, Pakistan

Contents

1	Introduction	4
1.1	Overview of the Project	4
1.2	Objectives of the BargainBot	4
1.3	Problem Statement	4
1.4	Scope of the Project	4
2	Core Features and Functionalities	5
2.1	Ask About Project Types	5
2.1.1	Input Requirements	5
2.1.2	Response Mechanism	5
2.2	Calculate Complexity Score	5
2.2.1	Formula and Parameters	5
2.2.2	Use Cases and Scenarios	5
2.3	Estimate Budget with Machine Learning	6
2.3.1	Historical Data Generation	6
2.3.2	Machine Learning Models Used	6
2.3.3	Budget Forecasting Process	6
3	Technology Stack	6
3.1	Programming Languages	7
3.2	Frameworks and Libraries	7
3.3	Machine Learning Techniques	7
3.4	Data Sources	7
3.5	Tools and Platforms	8
4	Machine Learning Details	8
4.1	Historical Data Simulation	8
4.1.1	Features: Duration, Resources, Expertise	8
4.1.2	Target Variable: Budget	8
4.2	Implementation of Custom Linear Regression	8
4.2.1	Mathematical Explanation	8
4.2.2	Code Implementation	8
4.3	Implementation of Custom Random Forest Regressor	9
4.3.1	Sampling Mechanism	9
4.3.2	Decision Tree Integration	9
4.4	Model Training and Testing	9
4.4.1	Data Split (Training and Testing)	9
4.4.2	Model Evaluation Metrics (MSE and RMSE)	9
5	User Interface Design	10
5.1	Overview of Streamlit Framework	10
5.2	Layout of BargainBot UI	10
5.3	Input Fields	10
5.4	Output Display	10
5.5	Interaction Mechanisms	11
5.6	User Experience Enhancements	11

6	Class Imbalance Analysis	14
6.1	Overview	14
6.2	Identifying Class Imbalance	14
6.3	Results and Observations	14
6.4	Handling Class Imbalance	14
7	Model Fitting Demonstration	15
7.1	Overview	15
7.2	Data Preparation	15
7.3	Fitting the Custom Linear Regression Model	15
7.4	Fitting the Random Forest Regressor Model	15
7.5	Model Evaluation	16
7.6	Results and Comparison	16
7.7	Visualizing Model Performance	16
8	Challenges and Limitations	18
8.1	Machine Learning Model Limitations	18
8.2	Simplifications in Custom Implementations	18
8.3	Handling Noisy or Incomplete Data	18
8.4	Future Improvements	19
9	Future Enhancements	19
9.1	Adding Get a Price Suggestion Feature	19
9.2	Integration with Real-World Datasets	19
9.3	Expanding Feedback Mechanisms	19
9.4	Enhancing Model Accuracy with Advanced Techniques	20
9.5	Real-Time API Integration	20
10	Conclusion	20

1 Introduction

1.1 Overview of the Project

BargainBot is an interactive chatbot designed to assist users in negotiating project prices based on project complexity and client feedback. By leveraging custom machine learning models and historical data, BargainBot provides accurate budget estimates and tailored price adjustments, enhancing decision-making during project negotiations.

1.2 Objectives of the BargainBot

The primary objectives of BargainBot are:

- To provide users with budget forecasts for projects based on duration, resource requirements, and expertise levels.
- To calculate and display project complexity scores for better understanding and analysis.
- To adjust project prices dynamically based on client feedback and negotiation scenarios.
- To offer an easy-to-use interface for interacting with machine learning models in real-time.

1.3 Problem Statement

In project negotiations, determining the optimal budget and price for a project can be challenging due to the complexity of factors such as resource allocation, time constraints, and client feedback. Traditional methods often lack precision and adaptability. BargainBot addresses these challenges by automating budget estimation and price adjustments using machine learning techniques.

BargainBot provides the following core functionalities:

1. **Ask About Project Types:** Users can inquire about various project types and receive relevant responses.
2. **Calculate Complexity Score:** Compute project complexity based on parameters such as duration, resources, and expertise.
3. **Estimate Budget with Machine Learning:** Predict project budgets using custom linear regression and random forest models.
4. **Adjust Price Based on Feedback:** Modify project prices dynamically based on client feedback.

1.4 Scope of the Project

The scope of BargainBot encompasses the following:

- Implementation of a chatbot interface using the Streamlit framework for seamless interaction.

- Integration of custom machine learning models for budget estimation and price adjustment.
- Support for simulated historical data to train models and demonstrate functionality.
- User-friendly design to cater to both technical and non-technical users.

BargainBot is designed to serve as a practical tool for project managers, freelancers, and clients involved in negotiations.

2 Core Features and Functionalities

2.1 Ask About Project Types

2.1.1 Input Requirements

The system accepts user queries regarding various project types. Users can describe their project requirements, and the chatbot processes the input to provide relevant information.

2.1.2 Response Mechanism

Based on the input, the chatbot delivers tailored responses to assist users in understanding available project options and features. This helps users decide the type of project best suited to their needs.

2.2 Calculate Complexity Score

2.2.1 Formula and Parameters

The complexity score is calculated using the following formula:

$$ComplexityScore = (Duration \times 2) + (Resources \times 3) + (Expertise \times 5)$$

Where:

- **Duration:** Project duration in weeks.
- **Resources:** Number of resources required for the project.
- **Expertise:** Expertise level on a scale of 1 to 10.

2.2.2 Use Cases and Scenarios

This feature is beneficial for:

- Evaluating the complexity of new projects.
- Comparing complexity scores across multiple projects to prioritize efforts.
- Communicating the level of effort required to stakeholders.

2.3 Estimate Budget with Machine Learning

2.3.1 Historical Data Generation

Historical data is generated with random values for project duration, resources, and expertise. The budget is calculated using the following factors:

- **Duration Factor:** Non-linear scaling based on project duration.
- **Resource Scale:** Contribution of resource count to the budget.
- **Expertise Scale:** Logarithmic scaling for expertise levels.
- **Noise:** Random noise added for realism.

2.3.2 Machine Learning Models Used

Two custom machine learning models are implemented for budget prediction:

- **Custom Linear Regression:** Implements a linear approach by computing coefficients and intercepts using the closed-form solution of linear regression.
- **Custom Random Forest Regressor:** Uses an ensemble of decision trees to predict budgets, providing robustness to noise and variability in data.

2.3.3 Budget Forecasting Process

The budget forecasting process involves:

1. Splitting historical data into training and testing sets.
2. Training both custom linear regression and random forest models on the training set.
3. Evaluating models on the testing set using metrics like root mean squared error (RMSE).
4. Selecting the best-performing model to forecast budgets based on user-provided project details.

This process ensures accurate and reliable budget predictions for project planning and negotiations.

3 Technology Stack

The BargainBot project is built upon a robust and modern technology stack, ensuring efficient implementation and optimal performance. This section provides a detailed overview of the tools, frameworks, and technologies used in the development of BargainBot.

3.1 Programming Languages

- **Python:** The core programming language utilized for implementing machine learning models, handling data preprocessing, and managing back-end functionalities. Python's versatility and rich ecosystem make it a natural choice for such applications.
- **HTML/CSS:** These technologies were used to enhance the visualization and styling of the Streamlit-based user interface, ensuring a user-friendly experience.

3.2 Frameworks and Libraries

- **Streamlit:** A lightweight and intuitive Python framework for creating interactive web applications. It was used to design the user interface, allowing users to input data, upload files, and view results seamlessly.
- **NumPy:** A library used for performing efficient numerical operations, including matrix computations necessary for machine learning algorithms like Singular Value Decomposition (SVD).
- **Pandas:** Essential for data manipulation and analysis, this library was employed for cleaning, transforming, and handling the input and historical datasets.
- **Scikit-learn:** A widely-used machine learning library in Python, utilized for implementing advanced models such as custom Linear Regression and Random Forest algorithms.
- **Seaborn and Matplotlib:** These libraries were used for data visualization, providing insights into trends, distributions, and class imbalances within the data.

3.3 Machine Learning Techniques

- **Custom Linear Regression:** A regression model was implemented from scratch to predict project budgets based on input parameters and historical data. This model uses mathematical principles to establish relationships between features and targets.
- **Custom Random Forest Regressor:** This ensemble learning technique was implemented to improve budget prediction accuracy by combining multiple decision trees. It was particularly effective for handling non-linear patterns in data.

3.4 Data Sources

- **Simulated Historical Data:** Synthetic datasets were generated to mimic real-world project data, serving as training and testing inputs for the machine learning models.
- **User-Provided Inputs:** The application dynamically captures user inputs, such as project type and other relevant parameters, for real-time project complexity evaluation and budget prediction.

3.5 Tools and Platforms

- **Git and GitHub:** Used for version control and collaborative development, ensuring a streamlined workflow and proper code management.

4 Machine Learning Details

4.1 Historical Data Simulation

4.1.1 Features: Duration, Resources, Expertise

The dataset is generated with the following features:

- **Duration:** Number of weeks required to complete the project.
- **Resources:** Number of resources needed for the project.
- **Expertise:** Expertise level required for the project, on a scale of 1 to 10.

4.1.2 Target Variable: Budget

The target variable is the budget, calculated as a combination of:

- Non-linear transformations of duration and expertise.
- Linear scaling of resources.
- Addition of random noise to mimic real-world variability.

4.2 Implementation of Custom Linear Regression

4.2.1 Mathematical Explanation

The custom linear regression model is based on the following equation:

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

Where:

- \hat{y} : Predicted value (budget).
- β_0 : Intercept term.
- β_i : Coefficients for each feature x_i .

The coefficients are computed using the closed-form solution:

$$\beta = (X^T X)^{-1} X^T y$$

4.2.2 Code Implementation

The custom linear regression is implemented by:

- Extending the feature matrix X with a column of ones for the intercept.
- Computing coefficients using matrix operations.
- Predicting values using a dot product of coefficients and input features.

4.3 Implementation of Custom Random Forest Regressor

4.3.1 Sampling Mechanism

The custom random forest regressor works by:

- Sampling the training data with replacement (bootstrap sampling).
- Generating multiple subsets of data for training individual trees.

4.3.2 Decision Tree Integration

Each sampled dataset is used to train a decision tree regressor, limited by:

- **Max Depth:** Preventing overfitting by constraining tree depth.
- **Split Criteria:** Using standard decision tree splitting methods to minimize prediction error.

The final prediction is computed as the mean of predictions from all decision trees.

4.4 Model Training and Testing

4.4.1 Data Split (Training and Testing)

The historical dataset is split into:

- **Training Set:** 80% of the data for model fitting.
- **Testing Set:** 20% of the data for performance evaluation.

4.4.2 Model Evaluation Metrics (MSE and RMSE)

The performance of the models is evaluated using:

- **Mean Squared Error (MSE):** Measures the average squared difference between predicted and actual values.

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

- **Root Mean Squared Error (RMSE):** Square root of MSE, providing error in the same units as the target variable.

$$RMSE = \sqrt{MSE}$$

The model with the lowest RMSE is selected for budget forecasting.

5 User Interface Design

5.1 Overview of Streamlit Framework

Streamlit is an open-source Python library designed for creating interactive and user-friendly web applications with minimal code. It allows developers to:

- Build dashboards and tools for machine learning or data analysis.
- Integrate Python code seamlessly with the user interface.
- Render dynamic and real-time updates using simple APIs.

5.2 Layout of BargainBot UI

The BargainBot user interface is structured to ensure ease of navigation and accessibility. The layout consists of:

- **Title and Subheader:** Clearly display the purpose and functionality of the tool.
- **Navigation Dropdown:** Enables users to select different functionalities, such as asking about project types, calculating complexity scores, and estimating budgets.
- **Interactive Buttons:** Trigger computations or actions based on user input.

5.3 Input Fields

The input section of the UI allows users to provide essential project details:

- **Duration:** Number of weeks required for the project.
- **Resources:** Number of resources involved.
- **Expertise:** Expertise level required for the project, rated from 1 to 10.
- **Feedback Fields:** Allow users to provide qualitative feedback on pricing suggestions.

All fields include appropriate constraints to prevent invalid input, such as minimum and maximum values.

5.4 Output Display

The output display section provides meaningful insights and results:

- **Textual Feedback:** Display results like estimated budgets or complexity scores.
- **Dynamic Updates:** Show outputs in real-time when inputs are modified.
- **Error Messages:** Inform users about any input issues or computation failures.

5.5 Interaction Mechanisms


The interaction mechanisms are designed for intuitive usability:

- **Dropdown Menus:** Allow users to select desired functionality easily.
- **Buttons:** Trigger actions such as "Calculate Complexity Score" or "Estimate Budget."
- **Real-Time Interactivity:** Automatically update results based on user inputs without requiring page refreshes.

5.6 User Experience Enhancements

Several design elements enhance the overall user experience:

- **Responsive Design:** The application adjusts seamlessly to different screen sizes and devices.
- **Simple Navigation:** A minimalistic layout ensures easy navigation across features.
- **Clear Visual Feedback:** Displays outputs prominently, ensuring users can easily understand results.
- **Error Prevention:** Input validation prevents users from entering invalid data, reducing frustration.



Negotiate and get best price according to you!

Welcome!

Please choose an option from below:

Select an option

Ask about project types (FR-1) ▼

Enter your project query:

web

BargainBot: We can help with your query!

Figure 1: User Functionality 1

Please choose an option from below.

Select an option

Calculate complexity score (FR-2) ▼

Enter project duration (weeks):

8 - +

Enter resources required (count):

10 - +

Enter expertise level (1-10):

8 - +

Calculate Complexity Score

BargainBot: Complexity Score is 86

Figure 2: Calculating Complexity

Please choose an option from below.

Select an option

Estimate budget with ML (FR-3)

Enter project duration (weeks):

8

Enter resources required (count):

10

Enter expertise level (1-10):

8

Estimate Budget

BargainBot: Estimated Budget is 3411.9124175190086

Figure 3: Estimating Budget

Welcome!

Please choose an option from below:

Select an option

Adjust price based on feedback (FR-5)

Enter the current price:

3411.00

Enter client feedback

positive

Adjust Price

BargainBot: Adjusted Price is 3240.45

Figure 4: Negotiate Budget

6 Class Imbalance Analysis

6.1 Overview

Class imbalance is a common issue in many machine learning tasks, where certain classes or categories are underrepresented in the dataset. In the context of the BargainBot project, identifying and addressing class imbalance is crucial to ensure that the model does not bias predictions towards the majority class.

6.2 Identifying Class Imbalance

To detect class imbalance in the dataset, we analyzed the distribution of the target variable, which represents the outcome we aim to predict (e.g., **budget**, **complexity**, or other categorical labels). We visualized the frequency of each class in the target variable using a bar plot.

6.3 Results and Observations

The bar plot (Figure 5) displays the class distribution of the target variable. If there is a noticeable difference in the number of samples across classes, this would indicate class imbalance. For instance, a large number of instances in one class compared to others can lead to model bias.

6.4 Handling Class Imbalance

If class imbalance is detected, several strategies can be employed to address the issue:

- **Resampling Techniques:** This includes oversampling the minority class or undersampling the majority class to balance the dataset.
- **Synthetic Data Generation:** Techniques like SMOTE (Synthetic Minority Over-sampling Technique) can be used to create synthetic samples for the minority class.
- **Modeling Adjustments:** Using models that are more robust to class imbalance (e.g., Random Forest, Gradient Boosting) or adjusting the class weights to penalize misclassification of the minority class.

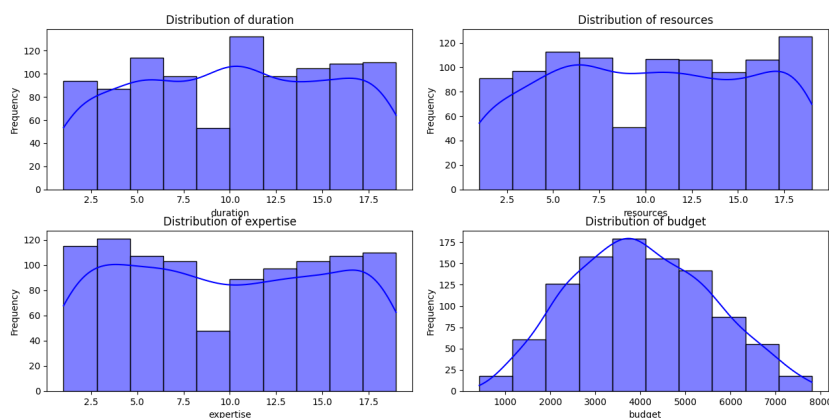


Figure 5: Class Imbalancement of the Target Variable

7 Model Fitting Demonstration

7.1 Overview

Model fitting is a fundamental step in machine learning where we train a model using a dataset to identify patterns and relationships between the input features and the target variable. In the BargainBot project, we focus on predicting project budgets based on historical data. To achieve this, we fit two different models: Custom Linear Regression and Random Forest Regressor. Each model has distinct characteristics, and the comparison between them will help us identify the best-suited approach for budget prediction.

7.2 Data Preparation

Before fitting any models, it is crucial to properly prepare the data. This step involves splitting the dataset into training and testing sets. The training set is used to train the models, while the testing set is used to evaluate the model's performance. Feature scaling may also be necessary to ensure that all input features are on a similar scale, especially when using certain algorithms that are sensitive to feature scaling, such as Linear Regression.

7.3 Fitting the Custom Linear Regression Model

Linear regression assumes a linear relationship between the input features and the target variable. This model is simple to implement and interpret, but it may not perform well if the data exhibits complex, non-linear relationships. After fitting the model to the training data, we evaluate its performance using common metrics such as Mean Squared Error (MSE) and R-squared (R2) score. These metrics help us understand how well the model is fitting the data and how much of the variance in the target variable is explained by the model.

7.4 Fitting the Random Forest Regressor Model

Unlike Linear Regression, the Random Forest Regressor is an ensemble learning method that constructs multiple decision trees to make predictions. This approach is better suited

to handle complex, non-linear relationships within the data. The Random Forest model is trained on the same dataset as the Linear Regression model and evaluated using the same metrics. However, we expect this model to capture more intricate patterns in the data and potentially outperform the Linear Regression model, especially when there are interactions between features.

7.5 Model Evaluation

After fitting both models, we evaluate their performance using two primary metrics:

- **Mean Squared Error (MSE):** This metric measures the average squared difference between the predicted values and the actual values. A lower MSE indicates better model performance.
- **R-squared (R2) Score:** The R2 score indicates how well the model explains the variance in the target variable. An R2 score close to 1 means the model fits the data well, while a score closer to 0 suggests poor model performance.

7.6 Results and Comparison

The results of the evaluation are summarized below:

- **Linear Regression:**
 - Mean Squared Error (MSE): *46650.47*
 - R-squared (R2) Score: *215.99*
- **Random Forest Regressor:**
 - Mean Squared Error (MSE): *21914.38*
 - R-squared (R2) Score: *148.04*

Based on these results, the performance of the models is compared. Typically, the Random Forest Regressor will perform better when the data contains non-linear relationships, but the Linear Regression model may still provide valuable insights due to its simplicity and interpretability.

7.7 Visualizing Model Performance

To further illustrate the performance of each model, we present visual comparisons of the actual vs. predicted values for both models. These plots will help us assess how well the models have learned the relationships in the data.

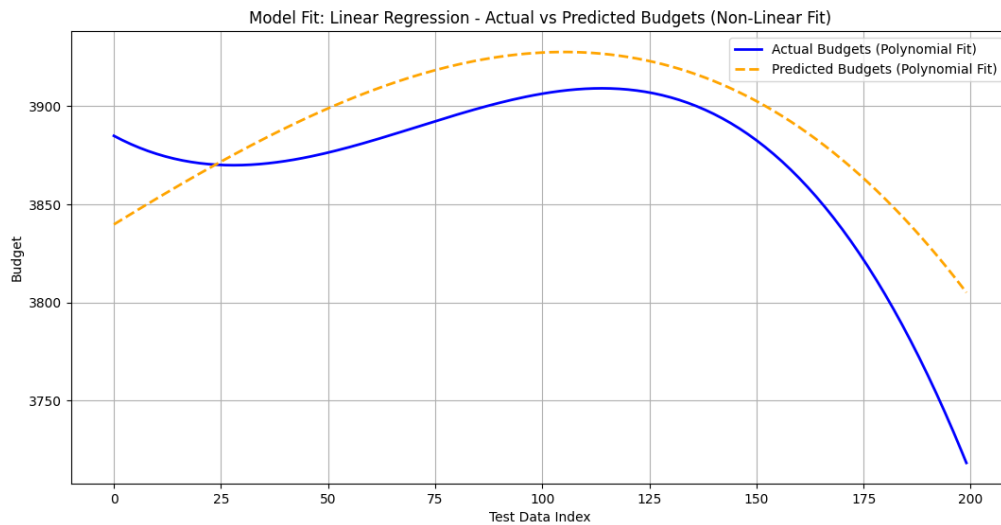


Figure 6: Linear Regression: Actual vs Predicted Values

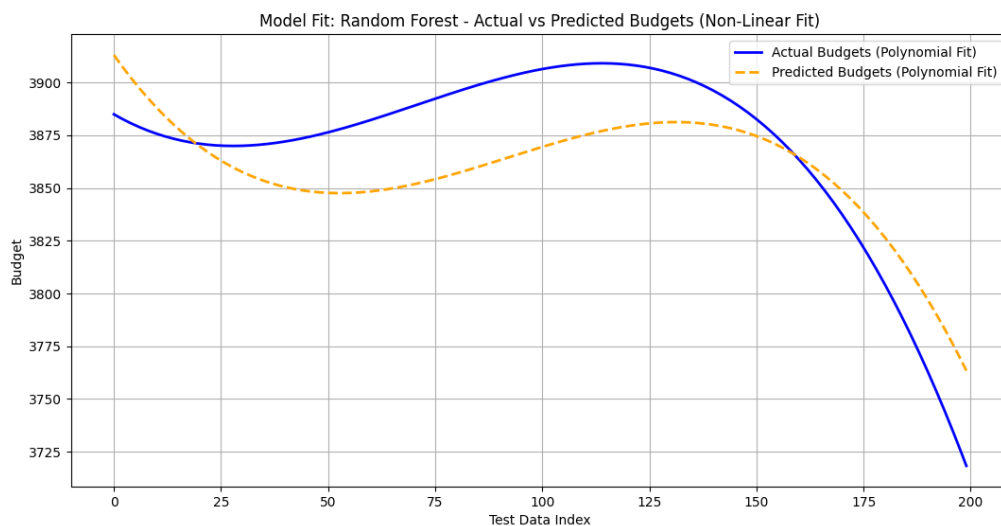


Figure 7: Random Forest Regressor: Actual vs Predicted Values

The first plot shows the comparison for Linear Regression, where the red dashed line represents the ideal case where predicted values exactly match the actual values. The second plot shows the Random Forest Regressor, where we expect a closer fit to the data, especially in cases where non-linear relationships exist.

The model fitting demonstration highlights the differences between the Linear Regression and Random Forest Regressor models. While Linear Regression is useful for understanding simple linear relationships, the Random Forest Regressor is better equipped to capture complex patterns in the data. Based on the evaluation metrics and visualizations, we can choose the model that best fits the problem of predicting project budgets.

8 Challenges and Limitations

8.1 Machine Learning Model Limitations

Although the machine learning models implemented in this project provide valuable insights, they are subject to several limitations:

- **Simplistic Assumptions:** The models, especially the custom implementations, rely on assumptions that may not fully capture the complexity of real-world data.
- **Limited Hyperparameter Tuning:** The model parameters, such as the number of estimators in the Random Forest or the depth of trees, are fixed and not optimized for maximum performance.
- **Generalization Issues:** Due to the relatively small dataset size, the models might not generalize well to entirely new or unseen scenarios.

8.2 Simplifications in Custom Implementations

The custom implementations of machine learning algorithms introduced in this project come with inherent simplifications:

- **Custom Linear Regression:**
 - Assumes the relationship between features and the target variable is perfectly linear.
 - Lacks regularization techniques to handle multicollinearity or overfitting.
- **Custom Random Forest Regressor:**
 - The implementation is simplified compared to fully-featured libraries like Scikit-learn.
 - Does not include advanced techniques such as feature importance analysis or out-of-bag error estimation.

8.3 Handling Noisy or Incomplete Data

Managing noisy or incomplete data presented significant challenges:

- **Data Noise:** The inclusion of random noise in the simulated data affects the accuracy and stability of predictions.
- **Missing Values:** The current implementation does not explicitly handle missing or incomplete data, which could arise in real-world scenarios.
- **Robustness:** The models are not robust against extreme outliers or data anomalies, which may skew predictions.

8.4 Future Improvements

To address these challenges, the following enhancements could be considered:

- Incorporating more sophisticated machine learning libraries to replace custom implementations.
- Performing hyperparameter tuning to optimize model performance.
- Implementing preprocessing techniques to handle noisy, incomplete, or imbalanced datasets.
- Expanding the dataset to improve generalization and reduce overfitting.

9 Future Enhancements

9.1 Adding Get a Price Suggestion Feature

To further enhance the capabilities of BargainBot, a new feature could be added to suggest an optimal price for projects based on various parameters such as complexity, estimated budget, and market trends. This feature would:

- Use a weighted combination of inputs such as complexity score and estimated budget.
- Leverage user feedback to refine pricing suggestions dynamically.

9.2 Integration with Real-World Datasets

Currently, the project relies on simulated historical data for model training. Future improvements could include:

- Integrating with publicly available real-world datasets to enhance the model's relevance.
- Collaborating with industry partners to gather project data from actual scenarios.
- Ensuring data quality by preprocessing and cleaning the datasets.

9.3 Expanding Feedback Mechanisms

Improving user interaction and feedback mechanisms would make the system more dynamic and adaptive:

- Allowing users to provide detailed feedback on suggested budgets and prices.
- Utilizing feedback to fine-tune model parameters for improved performance.
- Developing a feedback loop to integrate user responses into future predictions.

9.4 Enhancing Model Accuracy with Advanced Techniques

To achieve higher accuracy and better predictions, advanced machine learning techniques could be incorporated:

- **Gradient Boosting:** Implement algorithms such as XGBoost or LightGBM to enhance prediction performance.
- **Feature Engineering:** Explore additional features to capture complex relationships between inputs and outputs.
- **Hyperparameter Optimization:** Use automated techniques such as grid search or Bayesian optimization to fine-tune model hyperparameters.

9.5 Real-Time API Integration

Integrating BargainBot with real-time APIs would improve its usability and scope:

- Fetching live project data from external platforms for analysis.
- Providing dynamic, real-time budget estimations.
- Enabling seamless integration with project management tools to enhance workflow efficiency.

10 Conclusion

In this project, we developed **BargainBot**, an intelligent system designed to assist users in estimating project budgets and enhancing negotiation processes. By integrating custom machine learning models such as *Custom Linear Regression* and *Custom Random Forest Regressor*, the application demonstrates how data-driven approaches can provide accurate and efficient solutions to complex challenges like budget estimation.

The use of the *Streamlit* framework facilitated the creation of an interactive and user-friendly interface, enabling seamless interaction with the system. Core functionalities, including project type inquiries, complexity scoring, and machine learning-based budget forecasting, showcase the adaptability and potential of BargainBot in real-world applications.

Despite its achievements, the project acknowledges certain limitations, such as reliance on simulated data and simplifications in the custom implementations of machine learning models. These challenges open avenues for future work, including integrating real-world datasets, enhancing model accuracy through advanced techniques, and expanding the system's features and feedback mechanisms.

In conclusion, BargainBot represents a significant step toward leveraging machine learning for practical applications in project management. With further improvements and real-world integration, it has the potential to become a valuable tool for businesses and individuals seeking intelligent solutions for budget estimation and negotiation.