Milestone IV Report - Model Design and Utility Application
Predicting Used Car Prices using Machine Learning

CSCE 3602 - Fundamentals of Machine Learning

Omar Bahgat
Computer Engineering Department
The American University in Cairo
Cairo, Egypt
omar_bahgat@aucegypt.edu

Omar Saleh
Computer Engineering Department
The American University in Cairo
Cairo, Egypt
Omar_Anwar@aucegypt.edu

Introduction

In the previous phase of the project, we conducted a pilot study where we evaluated the performance of various models. In this phase, we will be establishing a design for our model and utility-application. This phase consists of two main parts. Firstly, it involves ensuring that the selected model is the most suitable for the nature of our data and the problem, alongside structuring the model and determining its optimal parameters such as the number of trees, max depth, and learning rate. Secondly, this phase includes the design of the utility application, which encompasses the application's architecture, user interaction mechanisms, and data flow.

Part I: Model Choice and Design

Model Choice

Based on the results of the pilot study conducted in the previous phase, the best performing models were XGBoost, Random Forests, and Artificial Neural Networks. Our final selection that best fits our data and problem was XGBoost as it yields the highest R^2 score as well as the fastest training time by a wide margin. Also, XGBoost has a wide range of parameters that we can fine tune to boost the performance of our model further. Several other factors which make XGBoost particularly useful for our dataset are the following:

Ensemble learning: XGBoost is an ensemble learner, combining predictions from multiple trees to improve model performance. This collective approach often leads to better predictions, particularly beneficial when tackling large datasets with several hundred features such as ours.

Mitigation of Overfitting: XGBoost effectively tackles overfitting, especially in complex datasets. By combining predictions from multiple trees, it maintains a balanced model. Additionally, techniques like random sampling during tree construction enhance its generalization.

Non-linear relationships: XGBoost's strength lies in its ability to capture complex nonlinear relationships between features and the target variable, a crucial advantage over linear models. By employing a powerful ensemble of decision trees, XGBoost can effectively model complex interactions and dependencies within the data, enabling it to uncover subtle patterns that may be missed by other models.

Model Design

In this part, we will outline the approach we will take to design our XGBoost model from scratch, focusing on its structure and its parameters.

Number of trees: To be able to approximate the optimal number of trees, we'll set a

conservative baseline by starting with a minimal count, around 5 trees and will gradually increase them. Our objective is to visualize the relationship between the number of trees and the model's effectiveness. We will achieve this by plotting the number of trees against the R^2 score and identify the point of diminishing returns, where additional trees cease to significantly enhance predictive power. This iterative process allows us to find a good balance between model performance and computational efficiency.

Max depth: The parameter *'max_depth'* determines the maximum depth of individual trees within the ensemble, significantly influencing model complexity and generalization. To effectively find an optimal max_depth value, we will initially set a conservative depth and adjust it, assessing its effects on model performance. Utilizing a visual representation such as a plot of max_depth against R^2 score, we aim to identify the optimal depth. This approach enables us to pinpoint the threshold where further increasing depth fails to significantly enhance performance, thus mitigating the risk of overfitting.

Learning rate: In our XGBoost model development, the *'learning_rate'* parameter holds significant importance, dictating the step size during each boosting iteration. We will concentrate solely on the constant learning rate strategy rather than the adaptive learning rate as it will greatly simplify our implementation. To find the optimal learning rate value, we will use the grid search method. This systematic approach involves exploring a range of values, allowing us to identify the most effective one to enhance our model's performance.

Min_child_weight: The *'min_child_weight'* parameter in XGBoost controls the complexity of individual trees and helps prevent overfitting by specifying the minimum sum of instance weights required in a child node. A higher min_child_weight results in more conservative tree growth, reducing the risk of capturing noise or outliers in the data. To choose the optimal value, we will use grid search, evaluating the model's performance with several different min_child_weight values and selecting the one that maximizes performance.

Reg_lambda: The parameter *'reg_lambda'* in XGBoost regulates model complexity by applying L2 regularization to the weights, preventing overfitting by penalizing large weight values. To find the optimal value, we will use grid search, testing different lambda values and selecting the one that maximizes the model performance.

Subsample: Subsampling in XGBoost involves randomly selecting subsets of the training data during tree construction, offering several advantages. It helps reduce overfitting, enhances generalization, and improves computational efficiency, while also enhancing regularization for more stable predictions. To determine the optimal subsample value, we will use grid search to experiment with different ratios and evaluate their impact on model performance, considering metrics like training and validation error, as well as computational efficiency. This iterative process will allow us to find the balance between reducing overfitting and maintaining model accuracy.

Part 2: Utility Application Design

Motivation

Our utility application uses a machine learning model for predictive analysis, trained on historical data to ensure accurate predictions. Our primary objective is to provide a user-friendly platform that predicts used car prices while integrating and learning from user feedback to enhance its predictions continuously. This empowers users to make informed decisions in the used car market,

offering valuable insights into the market value of their vehicles.

Architecture

The architecture of the utility application follows a client-server model, which offers advantages in terms of scalability, flexibility, and maintainability. Here's a breakdown:

Client-Server Architecture: The application is structured around a client-server paradigm, where distinct components handle specific functionalities. The client application serves as the interface through which users interact with the system, while the server component hosts the machine learning model responsible for generating predictions.

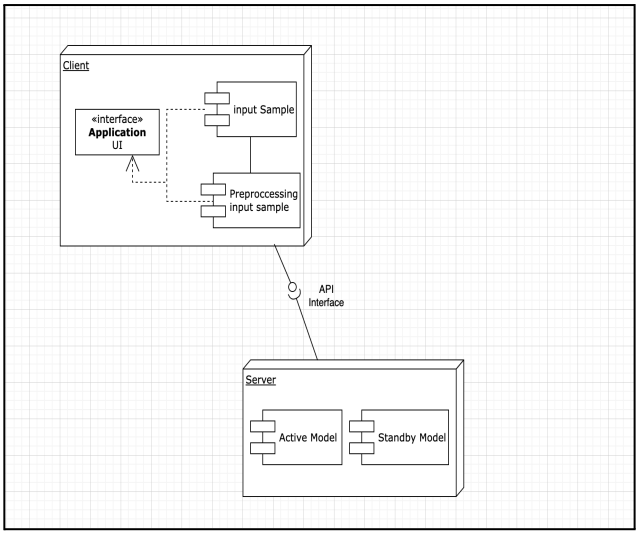
This architecture allows as to have:

Decoupled Components: One of the key features of this architecture is the decoupling of client and server components. This separation of concerns facilitates easier maintenance, updates, and scalability. Changes or enhancements made to one component do not necessarily impact the functionality of the other.

API Communication: Interaction between the client and server occurs through API communication. APIs provide a standardized mechanism for exchanging data between different software components, ensuring seamless integration.

Centralized Model Hosting: Hosting the machine learning model on the server enables centralized management and efficient processing of prediction requests from multiple users concurrently.

Here the illustration of the architecture as a component diagram



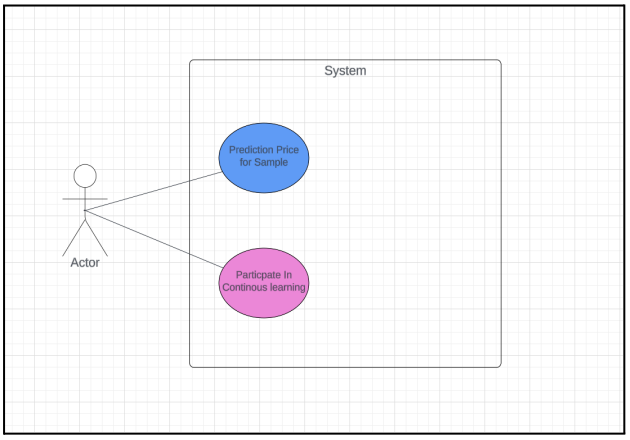
We will maintain two models on the server side so the application ensures continuous service availability even during model updates or retraining. While one model undergoes retraining or experiences issues, the other model remains active, ensuring uninterrupted prediction capabilities.

Human Interaction Flow

Users will be presented with options to input data for a car price prediction or participate in continuous learning.

Data Input: Through the user interface of the client application, users can input car-related data conveniently, utilizing various input methods such as text fields, drop-down menus, and selection boxes. To ensure the accuracy and reliability of the provided information, the application will employ robust data validation techniques. This includes real-time validation checks that verify the correctness of each input field, ensuring adherence to specified data formats, range constraints, and required fields. Additionally, the application will offer informative prompts and error messages to guide users in correcting any invalid or missing data entries promptly. By implementing thorough validation processes, the application maintains data integrity and enhances the overall user experience.

Continuous Learning: Our application also supports continuous improvement by allowing online learning capabilities. Users are encouraged to actively contribute additional data, helping the model enhance its accuracy and relevance over time.



Data Flow

Input Data Preprocessing: The initial step involves preprocessing user-provided data directly on the client side, ensuring its readiness for further processing.

Communication via API: Following preprocessing, the data is seamlessly transmitted to the server through API communication channels.

Prediction Generation: Upon receipt, the server uses the XGBoost machine learning model we developed to analyze the processed data and generate accurate predictions. These predictions are then relayed back to the client for user access..

Online Learning: An integral aspect of our system is its capability for online learning. Additional user-provided data is used to retrain the model, support ongoing improvement and adaptation. This iterative process ensures that the model remains dynamic and relevant in evolving scenarios, enhancing its predictive capabilities over time.

Here the illustration of the flow of the data utilizing a flowchart