CAR PRICE PREDICTION USING MACHINE LEARNING WITH STREAMLIT

Submitted by

KIBUMBA SEMEI EMMANUEL
Roll No.: 012230241

In the partial fulfillment for the award of Degree of

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY

(SOFTWARE ENGINEERING)

Supervisor: SupervisorName

ISBAT UNIVERSITY
A CHARTERED UNIVERSITY

Faculty of Information and Communication Technology

ISBAT University

Lugogo Bypass, Kampala, Uganda

July 1, 2025

Final Year Project Report

Project Title: CAR PRICE PREDICTION USING MACHINE LEARNING WITH STREAMLIT

KIBUMBA SEMEI EMMANUEL

A report submitted in part fulfilment of the degree of

MSc.IT (SOFTWAREENGINEERING)

Supervisor: Supervisor Name

ISBAT UNIVERSITY
A CHARTERED UNIVERSITY

Faculty of Information and Communication Technology

ISBAT University

Lugogo Bypass, Kampala, Uganda

July 1, 2025

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

INTERNATIONAL BUSINESS, SCIENCE AND TECHNOLOGY UNIVERSITY

BONAFIDE CERTIFICATE

Certified that this project report entitled CAR PRICE PREDICTION USING MACHINE LEARNING WITH STREAMLIT is bonafide work of …. …. **Mr. KIBUMBA SEMEI EMMANUEL** …….. <NAME OF THE CANDIDATE>__" who carried out the project work under my supervision, bearing Roll No. "<Roll>" and is a bonafide student of INTERNATIONAL BUSINESS, SCIENCE AND TECHNOLOGY UNIVERSITY (ISBAT), in
the partial fulfillment for the award of **Master of Science in Information Technology**

during **2023-2025**.

DEAN, FICT

…….

(Signature with date)

<Name of the Dean >

GUIDE

……..

(Signature with date)

<Name of the Guide >

External Examiner:

Name:

Signature:

Abstract
The proliferation of online car marketplaces and digital vehicle platforms has increased the demand for reliable tools that can predict used car prices based on various attributes. This project presents a data-driven solution using machine learning techniques to estimate car prices more accurately. Models such as Linear Regression, Random Forest, and XGBoost were evaluated using a dataset sourced from Kaggle. The model with the highest performance—Random Forest—was integrated into a user-friendly web application using Streamlit. The system allows users to input car details like year, mileage, fuel type, and ownership to receive a real-time price estimate in either USD or UGX. This project not only aids consumers and dealers in pricing decisions but also demonstrates how machine learning can enhance transparency and efficiency in the automotive resale market.

Key Words and Phrase
Car Price Prediction, Machine Learning, Random Forest, Streamlit App, Data Preprocessing, Regression, Feature Engineering, UGX Conversion, Ensemble Models, User Interface Design.

Computing Review Subject Codes
I.2.6 [Artificial Intelligence]: Learning—Parameter learning
I.5.2 [Pattern Recognition]: Design Methodology—Classifier design and evaluation
H.5.2 [Information Interfaces]: User Interfaces—Interaction styles
D.2.11 [Software Engineering]: Software Architectures—Data abstraction

Table of Contents (Preview)

1. …… Introduction and Background.

2. ……. System Functional Specification.

3. …….. System Performance Requirements.

4. …… System Design Overview.

5. ……. System Data Structure Specifications.

6. ……..… Module Design Specifications.

7. ……..…… System Verification.

8. ……..…….. Conclusions.

1. Introduction and Background

1.1 Statement of Problem Area (brief, non-technical)

In the current digital era, both consumers and car dealers rely heavily on online platforms to buy and sell vehicles. However, determining the fair market price for a used car remains a challenge due to inconsistent appraisal methods, regional price variations, and a lack of transparency. Buyers risk overpaying, while sellers may underprice due to uncertainty or lack of historical pricing data. Manual estimations often overlook crucial factors like vehicle depreciation, fuel type, ownership history, and market trends, which can significantly affect the car's true value. Therefore, there is a critical need for a reliable, intelligent, and user-friendly system that can provide accurate car price predictions using data-driven techniques.

1.2 Previous and Current Work, Methods and Procedures (representative)

Historically, car price estimation models have relied on simple linear or multiple regression methods, where the relationship between car attributes and price is assumed to be linear. These models, although easy to interpret, fail to handle nonlinear patterns, interactions among features, and data irregularities—common in real-world vehicle data.

Recent advancements in data science have brought about more sophisticated methods such as decision trees, ensemble learning techniques (Random Forest, XGBoost), and even deep learning approaches. Ensemble models, particularly Random Forest and XGBoost, have become the preferred choice for tabular datasets due to their robustness against overfitting, ability to model complex relationships, and better generalization to unseen data.

In current work, machine learning pipelines now include comprehensive preprocessing stages—handling missing values, encoding categorical data, and feature scaling—to optimize model performance. Additionally, tools like Streamlit and Flask have emerged to simplify the deployment of ML models into production-ready web applications, allowing real-time interaction and usability for end-users.

1.3 Background

With Uganda’s growing internet penetration and digital transformation, more automotive dealers and individual sellers are shifting their operations online. Despite this trend, price estimation tools that cater specifically to local or regional

contexts are lacking. Western-based models often ignore region-specific factors like currency differences (USD vs UGX), market demand, import taxes, and vehicle availability.

Car price prediction as a problem lies within supervised learning—specifically regression. The aim is to predict a continuous numeric value (price) based on various input features such as year of manufacture, kilometers driven, fuel type, transmission, and ownership type. By integrating these features into a machine learning model trained on historical data, we can estimate a fair market value for any given vehicle.

This project addresses this gap by developing a context-aware car price prediction model optimized for Ugandan and similar East African markets, making it not only accurate but also relevant to local users.

1.4 Brief Project Description (overview of new, extended or different functions, structure or operation)

The project entails the design, development, and deployment of a car price prediction application using machine learning. It begins by collecting and preprocessing a dataset of used cars, followed by training and evaluating multiple models—including Linear Regression, Random Forest, and XGBoost. The best-performing model, based on metrics such as RMSE and R^2 score, is integrated into a user-friendly web interface using Streamlit.

The application allows users to:

- Input car details via interactive UI widgets
- Select output in either **USD** or **UGX**
- Receive real-time price predictions based on model inference
- Visualize key performance indicators and model behavior (e.g., feature importance)

The system is designed to be extendable and customizable, with modular components that allow easy updating of models, UI, and datasets. It's suitable for use by dealerships, online car marketplaces, and individual sellers.

1.5 Purpose/Objectives/Justification of Project (theoretical, practical, or educational impacts on hardware, software, or users)

Theoretical Impact:

This project demonstrates how ensemble learning models can outperform traditional regression models in practical prediction tasks. It also offers insights into the effectiveness of preprocessing, feature engineering, and model tuning in improving prediction accuracy.

Practical Impact:

- Enables sellers to avoid underpricing their cars

- Helps buyers avoid overpaying or getting scammed
- Facilitates informed decisions in online car marketplaces
- Provides regional relevance with the inclusion of UGX as an output currency

Educational Impact:

- Serves as a real-world application of machine learning principles
- Demonstrates full-cycle ML project development—from data ingestion to model deployment
- Enhances the developer's knowledge of integrating ML with frontend tools like Streamlit

Justification:

The project not only fills a gap in the market for price estimation tools tailored to East Africa but also serves as a platform to bridge theoretical machine learning concepts with practical implementation and deployment. It aligns well with current industry needs and academic goals in data science, AI, and software engineering.

2. System Functional Specification

2.1 Functions Performed (itemize and describe)

No.	Function	Description
1.	Load Dataset	Reads car data from a CSV file and stores it in memory for processing.
2.	Preprocess Data	Handles missing values, encodes categorical variables, and normalizes data.
3.	Model Training	Trains multiple regression models (Linear Regression, Random Forest, XGBoost).
4.	Model Evaluation	Calculates metrics like RMSE, MAE, and R^2 Score to compare model performance.
5.	Price Prediction	Accepts user input and predicts price in real-time using the trained model.
6.	Currency Conversion	Converts predicted price into UGX using a fixed or live exchange rate.
7.	User Input Handling	Provides interactive widgets (dropdowns, sliders) to capture input data.
8.	Visualization	Displays data previews, model performance summaries, and feature importance.

2.2 User Interface Design

The web interface is built using Streamlit for simplicity, responsiveness, and accessibility. It features:

- A two-column layout for entering vehicle attributes.
- A sidebar for selecting output currency.
- Interactive widgets for each vehicle feature.
- A responsive "Predict Price" button.
- Display area for prediction results and optional performance charts.

2.2 Other User Input Preview

Field Name	Input Type	Description
Year	Slider	Year the car was manufactured
Fuel Type	Dropdown	Fuel type (Petrol, Diesel, etc.)
Seller Type	Dropdown	Dealer or Individual
Transmission	Dropdown	Manual or Automatic
Owner	Dropdown	Ownership history (1st, 2nd, etc.)
Kilometers Driven	Number	Total distance driven by the car
Currency Output	Toggle	Display result in USD or UGX

2.3 Other User Output Preview

Output Name	Description
Predicted Price	Estimated price of the car in USD or UGX
Evaluation Metrics	RMSE, MAE, R^2 (only visible in developer mode)
Feature Importance	Graphical view of factors influencing prediction

Output Name	Description

2.3 System Data Base/File Structure Preview

- Input File: Used Car Dataset.csv
- Temporary Memory Structure: Pandas DataFrame for holding preprocessed and encoded data
- Trained Model File (optional for deployment): Can be serialized using. Pkl format
- User Input Data: Handled in memory; not persisted

2.4 External and Internal Limitations and Restrictions

Type	Description
Internal	Limited to features available in the dataset (no image-based predictions)
Internal	Requires model retraining if dataset is updated
External	Accuracy may drop if applied to a different market or region
External	UGX conversion is fixed unless integrated with a live API
External	Prediction speed depends on user hardware and browser performance

2.5 User Interface Specification

2.6.1 Interface Metaphor Model

The interface follows a form-based metaphor, where users provide inputs as if filling out a form, then receive a computed output (price) upon clicking a submission button. It's intuitive and similar to existing forms used in car dealerships and websites.

2.6.2 User Screens / Dialog

Main Screen Includes:

- App Title and Description
- Input widgets in two-column layout
- Sidebar with currency toggle
- Button: “Predict Price”
- Output Box: Estimated price
- Optional: Feature importance chart and metrics in expandable sections

2.6.3 Report Formats / Sample Data

Example Input:

- Year: 2017
- Fuel Type: Petrol
- Transmission: Manual
- Seller Type: Dealer
- Owner: First Owner
- KMs Driven: 50,000

Example Output:

Estimated Price: 23,000,000 UGX

2.6.4 On-line Help Material

The app includes:

- Descriptions and tooltips for all widgets
- Optional expandable section: “How this model works”
- Markdown guide inside the app that explains:
    - What inputs mean
    - How the prediction is generated
    - How accurate the model is

2.6.5 Error Conditions and System Messages

Condition	Message
Missing user input	"Please fill in all fields before predicting."
Model loading failure	"Error: Could not load model. Please try again."
Invalid input type	"Input not recognized. Please enter valid data."
Prediction failure	"An error occurred during prediction."

2.6.6 Control Functions

Function	Description
Predict Button	Initiates prediction based on current inputs
Currency Toggle	Updates output currency (USD or UGX)
Expandable Sections	Reveals or hides advanced metrics and feature visuals
Sidebar Configuration	Let’s users adjust settings without affecting main UI

3. System Performance Requirements

3.1 Efficiency

The system is optimized for quick execution, offering real-time predictions with a response time of less than 1 second per request. The trained model is lightweight (<10 MB) and requires minimal RAM (<300 MB). The system does not require any external peripheral devices and operates efficiently on mid-range hardware (8GB RAM, Core i5 CPU).

3.2 Reliability

3.2.1 Reliability Measures

The system's reliability is measured using standard metrics:

- Accuracy (R^2 Score): > 85%
- Precision: Stable outputs for repeated inputs
- Consistency & Reproducibility: Same prediction under identical conditions

3.2.2 Error Detection and Recovery

The app handles input validation errors (e.g., missing or invalid values) with real-time user messages. If internal failures occur (e.g., model not loading), the app gracefully informs the user and halts operations without crashing. Logs can be implemented for error tracking during deployment.

3.2.3 Acceptable Error Rate

Acceptable RMSE is ≤ 10% of the average car price, with an expected system failure rate of <1%.

3.3 Security

3.3.1 Hardware Security

Runs locally or on trusted cloud servers. No physical security hardware needed.

3.3.2 Software Security

Runs in a sandboxed Python environment. No third-party data sharing or remote execution.

3.3.3 Data Security

No personal user data is stored. Input data is processed in-session and not logged or transmitted externally.

3.3.4 Execution Security

No login system included but can be extended with user authentication (e.g., for dealer platforms). Scripts are protected from code injection or misuse via

Streamlit's sandboxing.

3.4 Maintainability
Codebase is modular, readable, and well-commented. New features or datasets can be added without changing the entire architecture. Dependency management is minimal.

3.5 Modifiability
The system supports future updates like:
- Integration of new models (e.g., deep learning)
- Adding live exchange rate APIs
- Supporting additional countries or currencies

3. System Design Overview

4.1 System Data Flow Diagrams
At a high level, the system predicts car prices based on user inputs and presents the result in real time.

Level 0 (Context Diagram):
User → Enters car details → [Prediction System] → Displays predicted price

Level 1 (Data Flow Diagram):

Input Module – Collects user data

Preprocessing Module – Encodes and structures input data

Model Inference – Uses a trained ML model to predict price

Currency Converter – Converts USD prediction to UGX

Output Module – Displays the result to the user.

4.2 System Structure Charts

The system structure is modular:

Frontend Layer: Streamlit UI for user interaction

Middleware: Handles data preprocessing and user selections

Backend: Machine learning model for prediction and currency conversion

4.3 System Data Dictionary

Field Name	Description	Type
Year	Year of manufacture	Integer
Present_Price	Current market price (in Lakhs)	Float
Kms_Driven	Kilometers driven	Integer
Fuel_Type	Type of fuel	Categorical (encoded)
Seller_Type	Dealer or Individual	Categorical (encoded)
Transmission	Manual or Automatic	Categorical (encoded)
Owner	Number of previous owners	Integer

4.4 System Internal Data Structure Preview

Internally, data is stored and processed using a Pandas DataFrame before being passed to the ML model.
LabelEncoder is used for categorical encoding.
Final prediction-ready data is converted into a NumPy array for the model.

4.5 Description of System Operation (High Level)

1. User enters car details via Streamlit UI.
2. Data is encoded into numeric format (where applicable).
3. The trained Random Forest model receives inputs and predicts the price.
4. If selected, price is converted from USD to UGX using a preset exchange rate.
5. Final price is shown in real-time on the web interface.

4.6 Equipment Configuration

The system is designed to run on local or cloud infrastructure.

Minimum Requirements:

- CPU: Intel Core i5 or equivalent
- RAM: 4GB minimum (8GB recommended)

- OS: Windows / macOS / Linux
- Software: Python 3.8+, Streamlit, Scikit-learn, Pandas

4.7 Implementation Languages (Which and Why)

- **Python** was used due to:
    - Strong machine learning and data science ecosystem
    - Fast development and prototyping
    - Built-in support for web apps using Streamlit
    - Libraries like Scikit-learn, Pandas, Joblib

4.8 Required Support Software (Pre-existing)

The following tools/libraries are essential:

- **Python 3.8+**
- **Streamlit**: User interface development
- **Pandas**: Data handling and manipulation
- **Scikit-learn**: Model training and inference
- **Joblib**: (optional) for model persistence
- **NumPy**: Numerical computing

6. System Data Structure Specifications

5.1 Other User Input Specification

5.1.1 Identification of Input Data

User inputs required for car price prediction include:

- Year of manufacture
- Present price (in lakhs)
- Kilometers driven
- Fuel type
- Seller type
- Transmission type
- Number of previous owners
- Output currency (UGX or USD)

5.1.2 Source of Input Data (NOT input device)

Data is provided directly by the end user (car owner or dealer) based on the car's registration details, mileage, and known characteristics.

5.1.3 Input Medium and/or Device

Inputs are collected via a web interface (Streamlit form) accessed from a browser on desktop, tablet, or mobile devices.

5.1.4 Data Format/Syntax

All fields are structured, validated, and preprocessed:

Field	Format	Example
Year	Integer	2018
Present_Price	Float	5.5
Kms_Driven	Integer	40000
Fuel_Type	Categorical	"Petrol"
Seller_Type	Categorical	"Dealer"
Transmission	Categorical	"Manual"
Owner	Integer	0
Currency_Output	String	"UGX" or "USD"

5.1.5 Legal Value Specification

All input values are validated within expected and realistic ranges:

- Year: 2000 to current year
- Price: 0.1 to 50.0 lakhs
- Kilometers Driven: 5,000 to 200,000
- Owner: 0 to 3
- Fuel, Seller, and Transmission values must match the predefined categories

5.1.6 Examples

Example Input:

- Year: 2016
- Present_Price: 6.2
- Kms_Driven: 55,000
- Fuel_Type: Petrol
- Seller_Type: Dealer
- Transmission: Manual
- Owner: 0
- Currency_Output: UGX

5.2 Other User Output Specification

5.2.1 Identification of Output Data

- Predicted car price

- Optionally converted to UGX
- Internally: Model confidence or feature importance (for developers)
-

5.2.2 Destination of Output Data (NOT output device)
Displayed on the web page via Streamlit as formatted text or numeric output.

5.2.3 Output Medium and/or Device
Standard browser interface on any internet-enabled device (desktop, mobile, tablet).

5.2.4 Output Format/Syntax
- Displayed in a formatted currency style
    - UGX: UGX 25,000,000
    - USD: USD 6,500.00
- Optional graphical charts for advanced output

5.2.5 Output Interpretation (Meaning of Output)
The predicted value represents an estimated market price for the vehicle, calculated based on historical data and ML inference.

5.2.6 Examples
- Output 1: Estimated Price: UGX 27,450,000
- Output 2: Estimated Price: USD 7,210.50

5.3 System Database/File Structure Specification

5.3.1 Identification of Data Base/Files
- car_data.csv: Dataset used for training
- Trained model: In-memory or saved as model.pkl using Joblib (optional)
-

5.3.2 (Sub)systems Accessing the Data Base
- Model Training Module: Reads and processes car_data.csv
- Inference Engine: Uses the trained model file
- Streamlit UI: Triggers model load and prediction

5.3.3 Logical File Structure
- CSV: Structured with headers and tabular rows
- ML Model: Serialized binary format (.pkl)
- Files accessed using Pandas and Joblib libraries

5.3.4 Physical File Structure
- Location: Local directory or cloud server

- Access: Random access for file reads; streaming not required
- Organization: Files placed in the project root directory or /data/

5.3.5 Data Base Management Subsystems Used

- No external DBMS used
- Local CSV file storage and Pandas DataFrames serve as lightweight data storage for model input

5.3.6 Data Base Creation and Update Procedure

- Dataset updates require new car_data.csv
- Model must be retrained if significant data changes are made
- No automatic update mechanism (manual preprocessing and retraining required)

5.4 System Internal Data Structure Specification

5.4.1 Identification of Data Structures

- Pandas DataFrame: Used to store and preprocess the input data
- NumPy Array: Used for model prediction
- Encoders: LabelEncoder for categorical transformation

5.4.2 Modules Accessing Structures

- Preprocessing Module: Creates and encodes DataFrame
- Model Inference: Accepts NumPy array and returns predictions

Module Design Specifications

6.1 Module Functional Specification

6.1.1 Functions Performed

The application is broken into the following modules:

1. User Input Module
    - Captures user inputs via Streamlit widgets (year, mileage, price, etc.)
2. Preprocessing Module
    - Converts categorical values to numerical form
    - Normalizes and structures input for the model
3. Model Inference Module
    - Loads a pre-trained Random Forest model
    - Performs prediction based on processed inputs
4. Currency Conversion Module

    - Converts predicted price from base (INR/USD) to UGX
    - Uses a fixed exchange rate (e.g., 1 USD = 3800 UGX)
5. Output Display Module
    - Displays the final prediction to the user
    - Formats and styles the output appropriately in the UI

6.1.2 Module Interface Specifications (Input/Output)

Module	Inputs	Outputs
Input Collection	Streamlit widgets	Raw user input dictionary
Preprocessing	Raw input dict	Encoded NumPy array
Model Inference	NumPy array	Predicted car price
Currency Conversion	Price in USD	Price in UGX
Output Display	Price value	Rendered output on Streamlit interface

Global/Shared Variables:

- Exchange rate constant: USD_TO_UGX = 3800
- Trained model: loaded and cached once

6.1.3 Module Limitations and Restrictions

- Model accuracy is dependent on dataset quality
- Input validation is basic (cannot check real-world car specs)
- Currency conversion uses a fixed rate, not real-time API
- Only structured tabular input is supported (no image, VIN lookup, etc.)

6.2 Module Operational Specification

6.2.1 Locally Declared Data Specifications (Variable Dictionary)

Variable	Type	Purpose
year	int	Year of manufacture input
present_price	float	Current car price in lakhs
kms_driven	int	Distance driven
fuel_type	str	Fuel category ("Petrol", "Diesel", etc)

Variable	Type	Purpose
seller_type	str	Type of seller
transmission	str	Transmission type
owner	int	Number of previous owners
currency	str	Output currency ("UGX" or "USD")
prediction	float	Final predicted price

6.2.3 Description of Module Operation

1. User Input Module
    - Activated when Streamlit app is run
    - Captures values via st.slider(), st.selectbox(), st.radio()
2. **Preprocessing**
    - Encodes strings using LabelEncoder
    - Computes vehicle age as 2023 - year
    - Constructs model input array
3. **Model Inference**
    - Trained RandomForestRegressor is loaded and used
    - model.predict() returns numeric price estimate
4. **Currency Converter**
    - If user selects UGX, the predicted amount is multiplied by a fixed rate
    - USD remains unchanged
5. **Output Display**
    - Price is displayed using st.success() in readable currency format
    - Advanced users can expand to view model internals (optional)