AutoTrust AI: An Intelligent Vehicle Valuation and Trust Analysis Platform

Aditya Bawane[1], Girish Zade[2], Harshal Hivarkar[3], Sarang Khandate[4], Prof. Arpit Topre[5]
[1,2,3,4,5]*Department of Computer Science and Engineering (Data Science),*
Suryodaya College of Egineering and Technology, Nagpur, India

***Abstract*— AutoTrust AI is a web-based platform designed to support the valuation and trust assessment of used vehicles. The system combines a machine-learning-based price prediction module with vehicle information management, trust-related analysis, repair-cost information, PDF report generation, REST API support, and an analytics dashboard. The price prediction component uses a Random Forest Regression model trained using vehicle-related attributes such as manufacturing year, ownership history, mileage, insurance status, and repair-cost information. The platform is developed using Python and Django, with supporting libraries for data processing, machine learning, reporting, and visualization. The reported model evaluation values are a Mean Absolute Error (MAE) of 2141.69, Root Mean Squared Error (RMSE) of 4262.04, and coefficient of determination (R^2) of 0.8433. The system aims to make used-vehicle evaluation more structured and understandable. Details such as the dataset size, exact trust-score formula, and repair-cost calculation method were not specified in the available project description and should be documented after verification.**

***Index Terms*— AutoTrust AI, Used Vehicle Valuation, Machine Learning, Random Forest Regression, Trust Analysis, Django, REST API**

I. INTRODUCTION

The used-vehicle market involves decisions about vehicle condition, expected value, maintenance history, and reliability. Buyers may find it difficult to judge whether the asking price is reasonable or whether a vehicle has characteristics that require additional attention. AutoTrust AI proposes a web-based solution that brings vehicle information and machine-learning-based price estimation together in one platform. sUsers can enter vehicle details and receive a predicted price along with trust-related information and a generated report.

The project uses a Random Forest Regression model for price prediction and a Django-based web application for user interaction and data management. Additional modules provide PDF reports, analytics, and REST API access. The platform is intended as a decision-support tool; its output should not be treated as a substitute for a professional mechanical inspection.

II. PROBLEM STATEMENT

Used-vehicle buyers and sellers may not have access to a consistent way of estimating vehicle value from available vehicle attributes. Information about age, mileage, ownership, insurance, and repairs can be difficult to evaluate together. AutoTrust AI addresses this problem by providing a centralized platform that accepts vehicle details, predicts an estimated market value using a trained machine-learning model, and presents supporting trust-related information.

III. OBJECTIVES

- To develop a web-based system for collecting and managing used-vehicle information.
- To apply a Random Forest Regression model to estimate the price of a used vehicle from its available attributes.
- To provide trust-related analysis based on vehicle characteristics entered into the platform.
- To make vehicle evaluation results easier to review through downloadable PDF reports.
- To provide REST API endpoints for structured access to system functions and data.

- To present available analytical information through a dashboard.

IV. LITERATURE SURVEY

Breiman introduced Random Forests as an ensemble learning method that combines multiple decision trees to improve predictive performance and control overfitting [1]. Scikit-learn provides practical implementations of machine-learning algorithms, including Random Forest Regression, along with tools for preprocessing and model evaluation [2]. Django provides a Python web framework for building database-driven applications and managing user requests [3]. Django REST Framework extends Django with tools for building REST APIs [4].

The project also uses supporting technologies for reporting, visualization, and data processing. ReportLab can be used to generate PDF documents [6], Plotly supports interactive data visualization [7], and Pandas provides data structures and utilities for preparing tabular datasets [8]. YOLO-based object detection has been identified as a possible future extension for vehicle-damage analysis rather than a confirmed implemented module [5].

S. No.	Authors & Ref.	Focus Area / Work	Methodology / Key Technique	Contribution / Findings	Limitations / Gaps Addressed in AutoTrust AI
1	Breiman, L. (2001) [1]	Ensemble Learning Algorithms	Random Forest Regression & Decision Trees	Demonstrated that combining multiple decision trees improves predictive accuracy and mitigates overfitting in non-linear data regression.	Traditional model focused purely on theoretical regression without web/GUI integration or trust evaluation.
2	Pedregosa et al. (2011) [2]	Open-Source Machine Learning	Scikit-learn Framework in Python	Standardized implementations of machine learning models, feature selection, and evaluation metrics (MAE, RMSE, R^2).	Focused on core ML algorithms; lacks domain-specific integration like vehicle repair estimation or trust scoring.
3	Redmon et al. (2016) [5]	Real-Time Object Detection	Computer Vision using YOLO	Efficient detection of visual features and structural damage from digital imagery in real time.	Operates independently as an object detector; not natively integrated into vehicle valuation or price prediction workflows.
4	Existing Online Valuation Systems	Web-Based Vehicle Price Estimation	Rule-based valuation and basic Linear Regression	Offers rudimentary market price estimates based on basic parameters (age, brand).	Ignores multi-factor trust scores, detailed repair-cost estimation, downloadable PDF reports, REST APIs, and visual analytics.
5	AutoTrust AI (Proposed)	Integrated Valuation & Reliability Analysis	Random Forest + Django MVT + REST API + Plotly + ReportLab	Provides an accurate price prediction (R^2 = 0.8433), automated trust score calculation, repair estimation, interactive analytics, and downloadable reports.	Future scope includes full integration of image-based damage analysis via YOLOv8.

V. PROPOSED SYSTEM

AutoTrust AI is proposed as an integrated web platform with a user-facing interface and a Python-based backend. The user registers or logs in, enters vehicle information, and submits the details for evaluation. The backend stores the submitted information, prepares the input features, and passes them to the trained Random Forest Regression model. The system displays the predicted vehicle price and

trust-related information. Supporting modules provide repair-cost information, downloadable PDF reports, API responses, and dashboard visualizations.

The platform follows a modular structure so that authentication, vehicle data management, prediction, reporting, and analytics can be maintained separately. The exact trust-score calculation and repair-cost estimation logic should be described in the final technical documentation once the implemented rules or algorithms are confirmed.

VI. SYSTEM ARCHITECTURE AND WORKFLOW

The application follows Django's Model-View-Template (MVT) architectural pattern. The model layer manages stored information, the view layer processes requests and coordinates application logic, and the template layer presents the interface to users. The machine-learning model is integrated into the backend prediction workflow.

- User registration and login: the user accesses the platform through the authentication interface.
- Vehicle information entry: the user submits the available details of the vehicle.
- Database storage: the application stores the submitted vehicle information.
- Input preparation: the backend applies the required preprocessing to the submitted attributes.
- Price prediction: the prepared features are passed to the trained Random Forest Regression model.
- Trust-related analysis: the platform evaluates the available vehicle characteristics using its trust-analysis component.
- Report generation: the system prepares a PDF report containing the available evaluation results.
- API and dashboard: results can be accessed through REST API endpoints and summarized through dashboard visualizations.

VII. TECHNICAL STACK

Component	Technology / Tool
Backend	Python, Django
Frontend	HTML, CSS, Bootstrap
Database	SQLite
Machine Learning	Scikit-learn, Random Forest Regression
Data Processing	Pandas, NumPy
API Development	Django REST Framework
PDF Reporting	ReportLab
Visualization	Plotly
Version Control	Git and GitHub
Future Extension	YOLOv8 for vehicle-damage detection

VIII. RESEARCH METHODOLOGY

8.1 Data Collection and Input Attributes

The system uses vehicle-related attributes as input for price estimation. The available project description identifies manufacturing year, ownership history, mileage, insurance status, and repair-cost information among the relevant attributes. The complete dataset schema, source, size, and collection procedure should be added from the actual dataset used in the project.

8.2 Data Preprocessing

Before training, the dataset should be checked for missing values, inconsistent entries, duplicate records, and unsuitable data types. Categorical attributes must be converted into a numerical representation compatible with the selected model. The same preprocessing steps must be applied to new vehicle inputs so that the prediction model receives features in the expected format.

8.3 Model Training

The project uses Random Forest Regression to estimate vehicle prices. The dataset is used to train the model, and model performance is evaluated using regression metrics. The exact train-test split, hyperparameter settings, and validation procedure were not specified in the source project description and should be inserted from the actual experiment records.

8.4 Evaluation Metrics

Mean Absolute Error measures the average absolute difference between actual and predicted prices. Root Mean Squared Error gives greater weight to larger prediction errors. The coefficient of determination (R^2) indicates how much variation in the target values is explained by the model.

IX. IMPLEMENTATION PLAN

The implementation of AutoTrust AI is organized into sequential phases. Each phase produces a functional part of the application and is integrated with the other modules after its basic operation is verified. This plan describes the development activities and their expected outputs without assuming that unverified features have already been completed.

9.1 Requirement Analysis and Project Planning

The first phase is to identify the main users of the platform and the information they need during used-vehicle evaluation. The project requirements should be divided into functional requirements, such as user registration, vehicle-detail submission, price prediction, trust-related analysis, PDF report generation, API access, and dashboard display. Non-functional requirements such as usability, response time, data consistency, maintainability, and basic security should also be listed. The input attributes required by the machine-learning model must be finalized at this stage so that the web form, database, and model use a consistent data structure. The output of this phase is a documented requirement list and a module-wise development plan.

9.2 Development Environment and Technology Setup

The development environment is prepared by installing Python and creating a project-specific virtual environment. Django is installed for backend development, while Scikit-learn, Pandas, and NumPy are installed for data preparation and machine learning. Django REST Framework is included for API development, ReportLab for PDF generation, and Plotly for dashboard visualizations. SQLite is configured as the initial database. The source code is organized into separate Django applications or modules so that authentication, vehicle records, prediction, reporting, and analytics do not become mixed into one large program. Git can be used to maintain code history and track changes.

9.3 Dataset Preparation and Verification

The dataset used for vehicle-price prediction is collected from the project's available data source and inspected before model development. The columns, target price field, data types, and units are verified. Records with missing or invalid values are examined, and a suitable treatment is selected according to the actual data. Categorical columns are encoded, numerical columns are checked for inconsistent units, and duplicate or irrelevant records are handled. The dataset is then divided into training and testing portions using the project's selected evaluation procedure. The dataset source, number of records, selected attributes, and split ratio should be recorded in the final implementation report.

9.4 Machine-Learning Model Development

A Random Forest Regression model is trained using the prepared vehicle dataset. The model learns relationships between vehicle attributes and their corresponding prices by combining predictions from multiple decision trees. Model settings are selected and tested using the training data, while the held-out test data is used to evaluate predictive performance. The preprocessing steps and trained model are saved together or documented so that the same transformations can be applied when a user submits a new vehicle. The model file is then made available to the Django backend. The exact model parameters and training configuration should be reported from the actual code rather than estimated.

9.5 Backend Project and Database Development

The Django project is configured with the required settings, URL routes, templates, and database models. Database tables are designed for the information the application actually needs to retain, such as user accounts and submitted vehicle records. The model fields should match the vehicle attributes accepted by the input form and required by the prediction model. Database migrations are executed and tested to confirm that records can be created, retrieved, updated, and managed correctly. Input validation is added to prevent empty, incorrectly formatted, or out-of-range values from entering the prediction workflow.

9.6 User Registration and Authentication Module

A registration and login interface is developed so that users can access the platform through individual accounts. The application uses Django's authentication facilities to manage account credentials and login sessions. Registration fields and validation rules are defined according to the project requirements. Protected pages are made accessible only to authenticated users where required. Error messages

are displayed in a clear manner when registration details are incomplete or login credentials are invalid. This module provides the entry point for users before they submit vehicle information.

9.7 Vehicle Information Entry Module

A vehicle registration form is created to collect the attributes required for evaluation. Based on the available project description, these include manufacturing year, ownership history, mileage, insurance status, and repair-cost information. Additional fields should only be included when they are supported by the dataset and the implemented model. The form validates required fields, numeric values, and acceptable ranges before saving the record. A consistent mapping between form fields, database fields, and model input features is maintained to avoid incorrect predictions caused by missing or reordered attributes.

9.8 Price Prediction Integration

The price prediction module connects the trained Random Forest Regression model with the Django backend. When a valid vehicle record is submitted, the backend retrieves the required values, applies the same preprocessing used during model training, and arranges the features in the expected order. The model generates an estimated vehicle price, which is returned to the application and displayed to the user. The prediction result can also be associated with the corresponding vehicle record for later viewing and reporting. The application should handle invalid input or model-loading errors without displaying misleading results.

9.9 Trust Analysis Module

The trust-analysis module presents an assessment based on the vehicle characteristics available to the system. Its implementation should use the actual rules, factors, or model defined in the project code. The project description indicates that a trust score is part of the platform, but it does not provide the exact formula, factor weights, score range, or validation method. These details must therefore be documented from the implemented system before the score is described as a verified measure. The interface should explain that the result is a decision-support indicator and not a guarantee of vehicle condition or ownership history.

9.10 Repair-Cost Information Module

The platform includes a repair-cost-related component intended to support the vehicle evaluation process. During implementation, the project team should confirm whether repair cost is entered by the user, retrieved from stored information, estimated by a defined rule, or predicted by a separate model. The user interface and paper must describe only the method that is actually implemented. If repair cost is provided as an input attribute to the price model, the system should validate and store it consistently. If an estimation algorithm is used, its inputs, assumptions, and calculation procedure should be documented and tested.

9.11 PDF Report Generation

A report-generation module is integrated to produce a downloadable PDF containing the vehicle details and the evaluation outputs available for that record. ReportLab can be used to create the document layout, add headings and tables, and place the predicted price and trust-related result in clearly labelled sections. The report should identify the vehicle record and the date of generation where those details are available. Before the PDF is provided to the user, the application checks that the relevant record and prediction result exist. The report should not include fields or conclusions that were not generated by the application.

9.12 REST API Development

Django REST Framework is used to expose selected application functions through REST API endpoints. The API can accept validated vehicle data and return structured responses, such as a prediction result and associated evaluation information, according to the endpoints implemented in the project. Serializers are used to validate and convert application data into JSON-compatible representations. API routes, request methods, required fields, response formats, and error messages should be documented. Access control should be applied to endpoints that expose user-specific records.

9.13 Analytics Dashboard Development

The analytics dashboard presents summaries and visual representations of the data available in the application. Plotly can be used to display charts that help users or administrators understand stored vehicle records and prediction-related information. The

dashboard should use values retrieved from the database or actual model outputs rather than sample values presented as real project results. Each chart should have a clear title, axis labels, and units where applicable. The dashboard is tested to ensure that it updates correctly when new records are added.

9.14 Vehicle Image and Damage Detection as Future Work

The project description identifies YOLO-based vehicle-damage detection as a future enhancement. This capability should not be presented as a completed implementation unless an image dataset, trained or configured detection model, image-upload workflow, and evaluation results are available. In a future phase, users could upload vehicle images and a YOLO-based model could be investigated for detecting visible damage. Any detected damage would need careful validation and should not be treated as a complete mechanical inspection.

9.15 System Integration, Testing, and Deployment Preparation

After individual modules are developed, they are connected through the main application workflow. Integration testing checks the movement of information from the vehicle form to the database, preprocessing component, prediction model, trust-analysis module, report generator, API, and dashboard. Functional testing verifies registration, login, data submission, prediction display, report download, and API responses. Model testing uses the recorded evaluation procedure and metrics. Security and usability checks review input validation, access restrictions, error handling, and page navigation. Before deployment, the project team should configure environment-specific settings, protect secret keys, verify database configuration, and test the application in the intended hosting environment.

9.16 Expected Outcome of the Implementation Plan

On completion, the planned system is expected to provide a working web application in which an authenticated user can enter vehicle information, obtain a machine-learning-based price estimate, view the trust-related output supported by the implemented logic, and access the available report and dashboard features. REST API endpoints are intended to provide structured access to selected functions. The final project documentation should include screenshots, dataset details, model configuration, test cases, and evidence for each feature that has actually been completed.

X. EXPERIMENTAL RESULTS AND PERFORMANCE ANALYSIS

The project description reports the following evaluation values for the Random Forest Regression model. The currency and scale of the target price were not specified in the available description; therefore, the error values are reproduced without assigning a currency.

Table 2: Performance Evaluation Metrics

Evaluation Metric	Reported Value
Mean Absolute Error (MAE)	2141.69
Root Mean Squared Error (RMSE)	4262.04
R-squared (R^2)	0.8433

The reported R^2 value of 0.8433 indicates that the model explains a substantial proportion of the variation in the target values in the evaluation data. MAE and RMSE describe prediction error in the target's units. A complete reproducibility record should also provide the dataset source and size, train-test split, preprocessing steps, model parameters, and the evaluation procedure.

XI. LIMITATIONS AND FUTURE SCOPE

11.1 Limitations

1. The prediction accuracy depends on the quality, completeness, and reliability of the vehicle dataset used for model training.
2. The dataset source, total number of records, and complete list of input attributes are not specified in the available project description.
3. The reported evaluation metrics do not include the dataset split ratio, model hyperparameters, or detailed validation procedure.
4. The predicted vehicle price is an estimate and may differ from the actual market price because of location, demand, vehicle condition, and other factors not represented in the model.
5. The exact formula, factor weights, and validation method used to generate the trust score are not documented in the available project details.

6. The repair-cost estimation method is not fully specified; its reliability cannot be assessed until the implemented calculation or prediction method is documented.
7. The system depends on the accuracy of the information entered by users. Incorrect or incomplete vehicle details may affect the generated results.
8. YOLO-based vehicle-damage detection is described as a future enhancement and should not be considered an implemented feature without a working model and evaluation evidence.

XII. FUTURE SCOPE

1. A larger and more diverse vehicle dataset can be collected to improve model training and make predictions more representative of different vehicle categories and market conditions.
2. The model can be evaluated with additional validation methods and regularly updated data to monitor changes in prediction performance over time.
3. The price-prediction module can be improved by testing suitable regression algorithms and comparing their performance using the same evaluation dataset and metrics.
4. The trust-analysis module can be strengthened by defining transparent scoring factors, documenting their contribution, and validating the results against verified vehicle inspection information.
5. The repair-cost component can be enhanced by using a clearly documented estimation method supported by reliable repair and maintenance data.
6. YOLO-based image analysis can be explored to identify visible vehicle damage from uploaded photographs after collecting a suitable image dataset and evaluating detection performance.
7. The platform can be extended to support additional vehicle categories, more detailed vehicle specifications, and region-specific market information.
8. The user interface, API security, access control, and deployment configuration can be improved to support reliable use by a larger number of users.

XIII. CONCLUSION

AutoTrust AI presents a web-based approach to used-vehicle valuation and trust-related analysis by combining a Random Forest Regression model with a Django application. The platform is designed to collect vehicle details, estimate vehicle price, present supporting evaluation information, and provide reporting, API, and dashboard capabilities. The reported model results are MAE 2141.69, RMSE 4262.04, and R^2 0.8433. Further documentation and validation of the dataset, trust-score logic, and repair-cost method will strengthen the reproducibility and reliability of the project. Vehicle-damage detection using YOLO is retained as a possible future enhancement.

REFERENCES

[1] L. Breiman, “Random forests,” Machine Learning, vol. 45, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.
[2] F. Pedregosa et al., “Scikit-learn: Machine learning in Python,” Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011. [Online]. Available: https://jmlr.org/papers/v12/pedregosa11a.html
[3] Django Software Foundation, “Django documentation.” [Online]. Available: https://docs.djangoproject.com/
[4] Encode OSS Ltd., “Django REST framework documentation.” [Online]. Available: https://www.django-rest-framework.org/
[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2016, doi: 10.1109/CVPR.2016.91.
[6] ReportLab, “ReportLab user guide.” [Online]. Available: https://www.reportlab.com/docs/reportlab-userguide.pdf
[7] Plotly, “Plotly Python graphing library documentation.” [Online]. Available: https://plotly.com/python/
[8] The panda’s development team, “pandas documentation.” [Online]. Available: https://pandas.pydata.org/docs/