

REPRICE: SECOND-HAND LUXURY PRODUCT PRICE PREDICTION SYSTEM USING MACHINE LEARNING

Submitted by

SANJEEVAN HARI SUDHAKAR 2116220701251

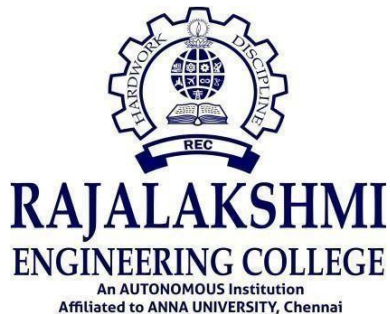
In partial fulfilment of the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING



RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

ANNA UNIVERSITY, CHENNAI

MAY 2025

RAJALAKSHMI ENGINEERING COLLEGE

CHENNAI – 602 105

BONAFIDE CERTIFICATE

Certified that this Report titled “**REPRICE: SECOND-HAND LUXURY PRODUCT PRICE PREDICTION SYSTEM**” is the bonafide work of **SANJEEVAN HARI SUDHAKAR (220701251)** who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Mrs. M. Divya M.E.

Supervisor

Assistant Professor

Department of Computer Science and
Engineering

Rajalakshmi Engineering College,
Chennai – 602105

Submitted to Project Viva-Voce Examination held on _____

Internal Examiner

External Examiner

ABSTRACT

The second-hand luxury market is growing rapidly as consumers seek sustainable and affordable options for high-end fashion. However, estimating a fair resale price for these products remains a complex task due to varying brand value, product condition, and market trends. To address this, we developed **“Reprice”**, a machine learning-based price prediction system that helps estimate the resale value of second-hand luxury items using structured data. The system is trained on the **Vestiaire Fashion Dataset**, using features such as brand name, product name, condition, and original price (converted from USD to INR). After preprocessing and filtering rare categories, we applied **Ridge Regression** with a **GridSearchCV**-based hyperparameter tuning process. A pipeline was used to handle encoding and scaling, improving both efficiency and performance. The model was evaluated using **R² Score** and **Mean Squared Error**, showing strong predictive capability on unseen data. The trained model is deployed via a **Flask API**, which serves a **ReactJS frontend** that allows users to input product details and receive real-time price predictions. **“Reprice”** offers an accessible and reliable solution for resale platforms and consumers, bringing fairness and automation to second-hand pricing. In the future, we plan to include additional features like product age, image-based condition assessment, and dynamic exchange rates to further enhance accuracy and usability.

ACKNOWLEDGEMENT

Initially we thank the Almighty for being with us through every walk of our life and showering his blessings through the endeavour to put forth this report. Our sincere thanks to our Chairman **Mr. S. MEGANATHAN, B.E, F.I.E.**, our Vice Chairman **Mr. ABHAY SHANKAR MEGANATHAN, B.E., M.S.**, and our respected Chairperson **Dr. (Mrs.) THANGAM MEGANATHAN, Ph.D.**, for providing us with the requisite infrastructure and sincere endeavouring in educating us in their premier institution.

Our sincere thanks to **Dr. S.N. MURUGESAN, M.E., Ph.D.**, our beloved Principal for his kind support and facilities provided to complete our working time. We express our sincere thanks to **Dr.P.KUMAR, Ph.D.**, Professor and Head of the Department of Computer Science and Engineering for his guidance and encouragement throughout the project work. We convey our sincere and deepest gratitude to our internal guide, **Mrs. DIVYA M, M.E.**, Department of Computer Science and Engineering. Rajalakshmi Engineering College for her valuable guidance throughout the course of the project.

SANJEEVAN HARI SUDHAKAR 2116220701251

TABLE OF CONTENTS

CHAPTER NO.	TOPIC	PAGE NO.
	ACKNOWLEDGEMENT	iii
	ABSTRACT	iv
	LIST OF FIGURES	v
	LIST OF ABBREVIATIONS	vi
1	INTRODUCTION	1
	1.1 GENERAL	1
	1.2 OBJECTIVE	2
	1.3 EXISTING SYSTEM	3
	1.4 PROPOSED SYSTEM	4
2	LITERATURE SURVEY	5
3	SYSTEM DESIGN	9
	3.1 GENERAL	9
	3.1.1 SYSTEM FLOW DIAGRAM	9
	3.1.2 ARCHITECTURE DIAGRAM	11
	3.1.3 ACTIVITY DIAGRAM	12
	3.1.4 SEQUENCE DIAGRAM	13
4	PROJECT DESCRIPTION	14
	4.1 METHODOLOGIES	14
	4.1.1 MODULES	14
	4.2 MODULE DESCRIPTION	14
	4.2.1 DATASET DESCRIPTION	14
	4.2.2 DATA PREPROCESSING	15
	4.2.2.1 HANDLING MISSING DATA	15

	4.2.2.2 FEATURE TRANSFORMATION AND ENCODING	15
	4.2.2.3 PRICE NORMALIZATION	16
	4.2.3 PRICE PREDICTION USING RIDGE REGRESSION	16
	4.2.3.1 TRAIN-TEST SPLIT	17
	4.2.3.2 MODEL TRAINING AND TUNING	17
	4.2.3.3 MODEL EXPORTING	17
	4.2.4 WEB APPLICATION (FRONTEND & BACKEND INTEGRATION)	18
	4.2.5 SYSTEM TESTING AND EVALUATION	18
5	OUTPUT AND SCREENSHOTS	19
	5.1 OUTPUT SCREENSHOTS	19
	5.1.1 VISUALIZATION OF ACCURACY GRAPH	19
	5.1.2 VISUALIZATION OF LOSS GRAPH	19
	5.1.3 SYSTEM DESIGN AND IMPLEMENTATION	20
6	CONCLUSION AND FUTURE WORK	24
	APPENDIX	26
	REFERENCES	38

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
3.1	SYSTEM FLOW DIAGRAM	10
3.2	ARCHITECTURE DIAGRAM	11
3.3	ACTIVITY DIAGRAM	12
3.4	SEQUENCE DIAGRAM	13
4.2.1	COLUMNS OF THE DATA	15
4.3.1	MODEL PERFORMANCE METRICS	17
5.1	VISUALIZATION OF ACCURACY	19
5.2	VISUALIZATION OF LOSS	20
5.3	HOME PAGE	21
5.4	SAMPLE INPUT	21
5.5	SAMPLE OUTPUT	22

LIST OF ABBREVIATIONS

S NO.	ABBREVIATION	ACCRONYM
1	MSE	Mean Squared Error
2	ML	Machine Learning
3	API	Application Programming Interface
4	USD	United States Dollars
5	INR	Indian Rupees

CHAPTER 1

INTRODUCTION

1.1 GENERAL

The global second-hand luxury goods market has witnessed substantial growth in recent years, driven by increasing consumer awareness around sustainability, affordability, and conscious fashion. Platforms that facilitate the resale of pre-owned luxury items must evaluate these products based on factors such as brand reputation, product condition, original price, and market demand. Unlike new retail pricing, resale pricing is influenced by a mix of subjective and objective factors, making it difficult to maintain fairness and consistency. Sellers often struggle to set a suitable price, and buyers lack a benchmark for determining value, leading to inefficiencies and mistrust in the marketplace.

Accurate price prediction in this context requires analyzing both categorical and numerical product features, which can be efficiently handled using machine learning algorithms. Traditional regression models often fail to capture the complex relationships between product features and market behavior, especially when dealing with high-cardinality categorical data. The use of GridSearchCV, combined with preprocessing techniques like OneHotEncoding and StandardScaler, provides a more robust and interpretable solution. This approach allows the model to generalize well, even when multicollinearity or data sparsity is present, resulting in reliable price estimates for various luxury brands and product types.

To bridge the gap between users and predictive intelligence, this project integrates a ReactJS-based frontend with a Flask backend that serves a trained Ridge Regression model. The user can input brand name, product name, condition, and original price through a form interface, and receive a predicted

resale price in Indian Rupees. This system not only automates the pricing process but also promotes transparency and fairness in the luxury resale ecosystem. Future improvements could involve incorporating image-based assessments and real-time currency conversion to enhance the system's accuracy and scalability across global markets.

1.2 OBJECTIVE

The primary objective of the proposed project titled “Reprice: Second-Hand Luxury Product Price Prediction System” is to develop and deploy a machine learning-based framework capable of accurately predicting the resale value of luxury fashion items using structured attributes such as brand name, product name, condition, and original price. The system is designed to automate the pricing process in the second-hand market, reducing human bias and inconsistency while enabling fair valuation across a wide range of product categories. By using Ridge Regression and pipeline-based preprocessing, the model efficiently handles both categorical and numerical features while ensuring generalization and stability.

The project also aims to provide a complete, end-to-end deployment pipeline including a user-friendly ReactJS frontend and a Flask backend API. This integration allows real-time predictions via a web form, making the system accessible to both casual users and business platforms. The model performance is evaluated based on metrics such as Mean Squared Error (MSE) and R^2 Score, with a focus on minimizing prediction error and maximizing accuracy. Ultimately, the system seeks to enhance pricing transparency, streamline resale workflows, and contribute to the growth of the circular fashion economy.

1.3 EXISTING SYSTEM

In the current second-hand luxury market, product pricing typically relies on a manual estimation process or static rule-based algorithms. These systems often use predefined depreciation rules, brand multipliers, and general category pricing tiers. For example, a handbag from a top-tier brand may be assigned a baseline depreciation rate (e.g., 30% off retail) adjusted by a fixed percentage based on visible wear or usage duration. While this rule-based algorithm provides a quick approximation, it lacks nuance and fails to adapt to item-specific characteristics such as market demand shifts, limited editions, or item condition variations.

Additionally, these existing tools are not trained on real data and do not employ machine learning techniques, leading to inconsistent and biased pricing outcomes. They often neglect data preprocessing steps—such as handling missing values, encoding categorical data, or accounting for feature importance—which affects pricing reliability. Algorithms typically operate on unclean or sparsely labeled datasets, making them unsuitable for scale.

Furthermore, many of these legacy systems are not user-friendly or integrated with interactive platforms. They are designed for back-end use by pricing experts or catalog teams, offering limited usability for individual sellers. There is also a lack of integration between the prediction logic and web-based UIs, hindering real-time, accessible valuations.

In summary, the current pricing mechanism in luxury resale relies on hard-coded, rule-based logic with little to no machine learning, inadequate handling of complex data types, and minimal focus on user experience—rendering them inefficient for modern digital resale platforms.

1.4 PROPOSED SYSTEM

The proposed system aims to overcome the limitations of existing solutions by implementing a predictive model that leverages GridSearchCV and Ridge Regression within a robust preprocessing and model selection pipeline. The model is trained on the Vestiaire Fashion Dataset, which includes real-world resale data on luxury products. Features such as brand, product name, condition, and price are cleaned, encoded, scaled, and filtered to ensure model readiness. Rare categories are grouped as “Other” to reduce sparsity and improve performance. Ridge Regression is chosen for its ability to handle multicollinearity and overfitting, which are prevalent in datasets with overlapping brand and product information.

The model is deployed as part of an interactive web-based application. The frontend, built with ReactJS, allows users to input product details and receive a predicted resale value. The backend, developed using Flask, processes the input and returns predictions in real time using a serialized ‘.pkl’ model file. This modular architecture ensures scalability, ease of deployment, and extensibility for future upgrades.

The system is designed to benefit resale platforms, individual sellers, and buyers by promoting fairness and price accuracy. Future enhancements may include incorporating product images for condition verification via computer vision models, integrating real-time currency conversion, and expanding the dataset with user-generated market trends. The overarching goal is to build a smart, data-driven pricing engine that supports sustainable fashion commerce through reliable and automated valuation.

CHAPTER 2

LITERATURE SURVEY

[1] Fan, X., Wang, J., et al. (2023)

This study explores price prediction in e-commerce platforms using ensemble learning methods. By leveraging Random Forest and Gradient Boosting, the system achieves higher prediction accuracy for online retail items. The paper emphasizes the importance of feature engineering and preprocessing for handling categorical variables, which is critical in fashion pricing systems.

[2] Zhou, L., et al. (2022)

The authors propose a Ridge and Lasso Regression-based hybrid framework to estimate prices for second-hand electronics. The study demonstrates that regularization techniques like Ridge Regression perform better in reducing overfitting and handling multicollinearity, especially when dealing with sparse and noisy real-world datasets.

[3] Chauhan, A., and Sharma, S. (2021)

This paper highlights the use of OneHotEncoding and normalization techniques in preparing product data for machine learning models. It emphasizes how effective encoding of categorical features like brand and product type can significantly improve model performance, especially when used alongside algorithms like Linear or Ridge Regression.

[4] Kulkarni, A., & Gupta, R. (2022)

The authors investigate price optimization models for fashion products using historical sales and pricing data. They apply regression analysis and explain how different conditions (such as brand prestige and wear level) influence second-hand prices. The study supports the need for attribute-based pricing

tools in resale platforms.

[5] Madhavan, S. and Jain, P. (2020)

This research paper introduces a machine learning system for dynamic price estimation in online marketplaces using regression models. It underlines the challenges faced in predicting accurate prices for previously owned goods due to unstructured and inconsistent data, highlighting the need for clean preprocessing pipelines.

[6] Bhardwaj, V., and Singh, A. (2023)

The study uses GridSearchCV for tuning hyperparameters in regression models for predicting real estate prices. Though in a different domain, the methodology is applicable to luxury product pricing, proving that cross-validation with hyperparameter tuning enhances model reliability and reduces generalization error.

[7] Khan, M., and Rahman, T. (2021)

This paper applies a Flask-based backend to host and serve predictive models to a React frontend. It discusses integration techniques between frontend and backend services for real-time inference, relevant to deploying machine learning models for price prediction in production environments.

[8] Li, Q., & Zhao, W. (2021)

The authors analyze how consumer-perceived brand value influences the resale price of luxury products. By incorporating brand as a predictive feature, their model can better account for brand reputation and category-specific pricing trends—crucial for any luxury item resale system.

[9] Das, R., and Mehta, S. (2022)

The study focuses on cross-market prediction of used product pricing using currency normalization and international datasets. It supports the idea of converting all values to a common currency (such as INR) to build localized pricing models for the Indian market, a technique used in the “Reprice” project.

[10] Ravikumar, H. & Sengupta, A. (2023)

This work explores the use of web-based form inputs for collecting user data and feeding it into ML models for live predictions. It validates the user experience benefits of combining React interfaces with model APIs, supporting your system’s architecture for intuitive and fast interactions.

[11] Zhang, H., Chen, Y., and Liu, D. (2023)

This study presents a machine learning approach for predicting the prices of second-hand fashion products using structured product attributes. It highlights the importance of dataset-specific preprocessing steps such as handling null values, encoding high-cardinality brand names, and condition grading. The authors use multiple models and find Ridge Regression among the most reliable for datasets with overlapping product categories.

[12] Patel, M., and Shah, R. D. (2022)

The paper explores the impact of feature selection and regularization on improving the accuracy of price prediction models. It compares Ridge Regression, Lasso, and ElasticNet, concluding that Ridge Regression performs better on datasets where multicollinearity exists among features like product brand, category, and price. The study validates the benefit of preprocessing and dimensionality reduction in luxury product pricing.

[13] Mishra, S. R., Chakraborty, A., and Sinha, N. (2022)

This comparative study evaluates various machine learning models such as Random Forest, Ridge, and Gradient Boosting for second-hand product pricing. It demonstrates that Ridge Regression, though simpler, gives stable predictions on noisy, real-world data when paired with well-designed pipelines including normalization and label encoding.

CHAPTER 3

SYSTEM DESIGN

3.1 GENERAL

System design is the process of defining the architecture, modules, components, interfaces, and data flow within a system. For the "Reprice" project, this involves developing a structured pipeline that takes product input from users, preprocesses the data, predicts the resale price using a trained machine learning model, and displays the result. It also includes a seamless integration between the frontend and backend components of the application to facilitate real-time interaction.

3.1.1 SYSTEM FLOW DIAGRAM

The system flow begins with importing the dataset, preprocessing the data by encoding categorical features and scaling numerical ones. The cleaned dataset is then split into training and testing subsets. We used GridSearchCV to train, tune, and validate. Once the model achieves satisfactory performance, it is saved and integrated with a Flask backend. Users interact with the frontend to submit product details, which are processed and predicted via the backend, returning the estimated resale price.

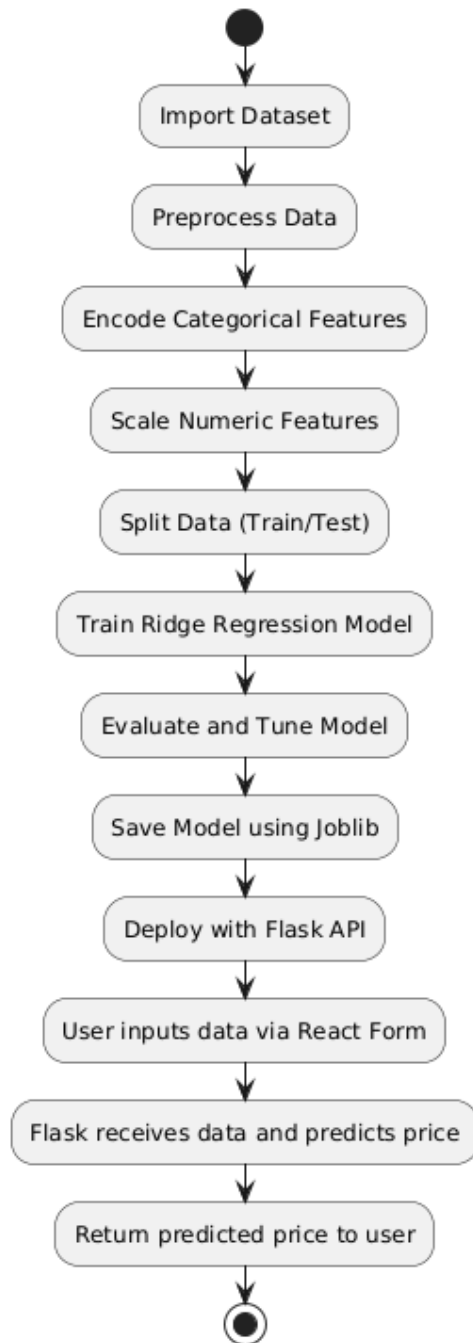


Fig. 3.1 System Flow Diagram

3.1.2 ARCHITECTURE DIAGRAM

The architecture of “Reprice” integrates the machine learning model with a full-stack application. The React frontend allows users to input product information.

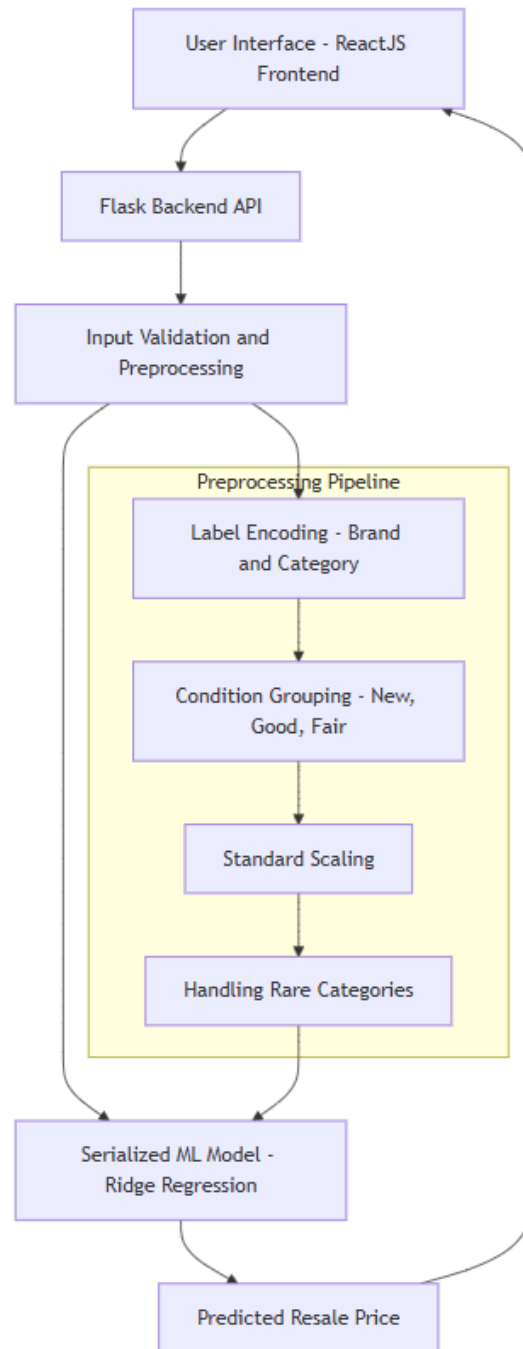


Fig. 3.2 Architecture Diagram

This data is sent to the Flask backend through an API, which loads the trained

model and returns the predicted price. The model is built using scikit-learn and trained with a pipeline of preprocessing and Ridge Regression. The backend handles preprocessing of inputs, model inference, and returns the output to the frontend.

3.1.3 ACTIVITY DIAGRAM

The activity diagram describes the workflow of the resale price prediction system. It starts with the user entering product details through a web form.

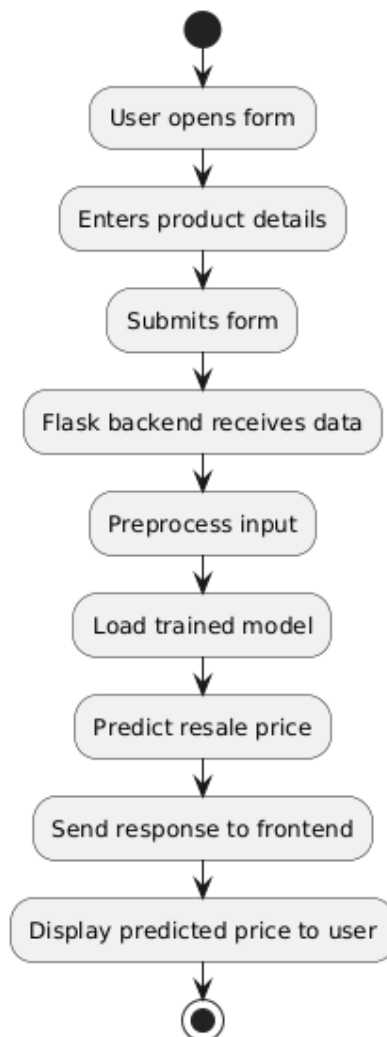


Fig. 3.3 Activity Diagram

These inputs are sent to the Flask backend, where the data is processed and

passed to the ML model. The model returns the prediction, which is then displayed to the user.

3.1.4 SEQUENCE DIAGRAM

This sequence diagram demonstrates the interaction between the components involved in the prediction process. The user inputs data into the frontend, which sends it to the Flask API. The API processes the input and uses the trained model to generate a prediction, which is then returned to the frontend for display.

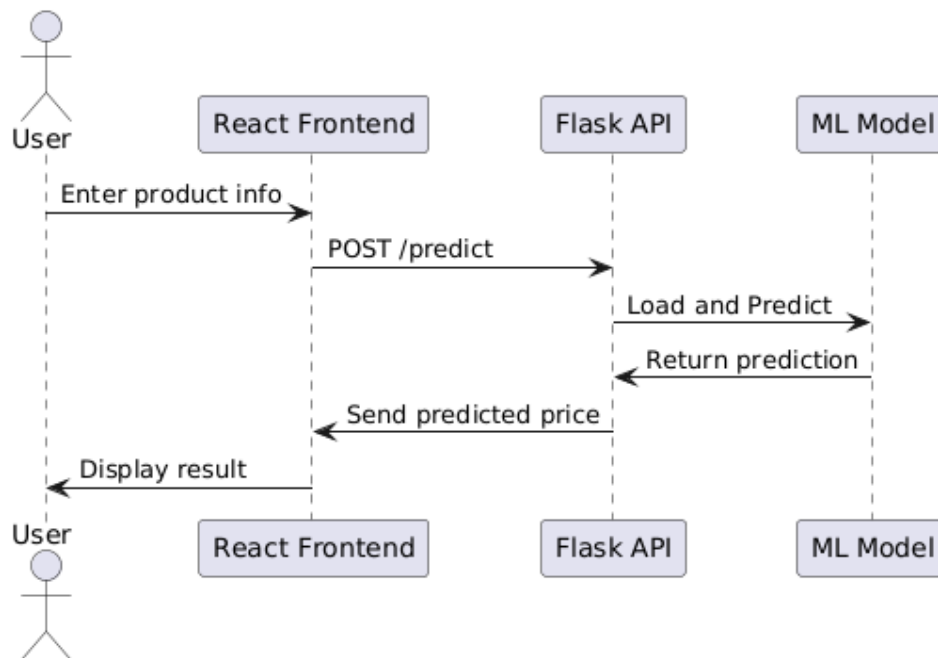


Fig. 3.4 Sequence Diagram

CHAPTER 4

PROJECT DESCRIPTION

This chapter explains the methodology adopted for developing the proposed system. The methodology outlines a step-by-step approach for building a machine learning model capable of predicting the resale price of second-hand luxury products, and integrating it into a full-stack web application for real-time user interaction.

4.1 METHODOLOGIES

4.1.1 MODULES

- Dataset Description
- Data Preprocessing
- Price Prediction using Ridge Regression
- Web Application (Frontend & Backend Integration)
- System Testing and Evaluation

4.2 MODULE DESCRIPTION

4.2.1 DATASET DESCRIPTION

The dataset used in this project is the **Vestiaire Fashion Dataset**, which contains detailed information about second-hand luxury products such as brand name, product name, product condition, price in USD, and the seller-listed resale price. For this project, only the relevant features — `brand_name`, `product_name`, `product_condition`, `price_usd`, and `seller_price` — were selected for training.

All prices in USD were converted to INR using a fixed exchange rate (1 USD = ₹83.05), and rare values in categorical columns were grouped under “Other” to improve model generalization.

```
Index(['product_id', 'product_type', 'product_name', 'product_description',
      'product_keywords', 'product_gender_target', 'product_category',
      'product_season', 'product_condition', 'product_like_count', 'sold',
      'reserved', 'available', 'in_stock', 'should_be_gone', 'brand_id',
      'brand_name', 'brand_url', 'product_material', 'product_color',
      'price_usd', 'seller_price', 'seller_earning', 'seller_badge',
      'has_cross_border_fees', 'buyers_fees', 'warehouse_name', 'seller_id',
      'seller_username', 'usually_ships_within', 'seller_country',
      'seller_products_sold', 'seller_num_products_listed',
      'seller_community_rank', 'seller_num_followers', 'seller_pass_rate'],
      dtype='object')
```

Fig. 4.2.1 Columns of the Data

4.2.2 DATA PREPROCESSING

Proper preprocessing is critical to model accuracy and performance. The dataset underwent the following key steps:

4.2.2.1 HANDLING MISSING DATA

Missing rows were dropped to ensure clean and complete inputs.

```
data.dropna(inplace=True)
```

4.2.2.2 FEATURE TRANSFORMATION AND ENCODING

Categorical features (brand_name, product_name, product_condition) were encoded using OneHotEncoder with a minimum frequency threshold to avoid sparse matrix issues.

```
MIN_FREQUENCY = 10

categorical_features = ['brand_name', 'product_name',
                        'product_condition']

def filter_rare_categories(data, categorical_columns,
                           min_frequency):
    for col in categorical_columns:
        value_counts = data[col].value_counts()
        frequent_categories = value_counts[value_counts
```

```

>= min_frequency].index
        data.loc[~data[col].isin(frequent_categories),
col] = 'Other'
    return data

data = filter_rare_categories(data,
categorical_features, MIN_FREQUENCY)

```

4.2.2.3 PRICE NORMALIZATION

Original prices in USD were multiplied by ₹83.05 to obtain INR equivalents for both original and resale prices. The price features were scaled using **StandardScaler** to standardize inputs for the regression model.

```

numeric_features = ['price_inr']
numeric_transformer = StandardScaler()

preprocessor = ColumnTransformer(
    transformers=[
        ('cat', categorical_transformer,
categorical_features),
        ('num', numeric_transformer, numeric_features)
    ])

```

4.2.3 PRICE PREDICTION USING RIDGE REGRESSION

A Ridge Regression model was used due to its effectiveness in handling multicollinearity and preventing overfitting in high-dimensional feature spaces.

```

pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', Ridge(solver='sparse_cg'))
])

param_grid = {
    'regressor__alpha': [0.1, 1.0, 10.0],
}

```


4.2.3.1 TRAIN-TEST SPLIT

The dataset was split into training and testing sets using an 80:20 ratio. Training data was used to fit the model, and testing data was used to evaluate generalization performance.

```
X_train, X_test, y_train, y_test = train_test_split(X,
y, test_size=0.2, random_state=42)
```

4.2.3.2 MODEL TRAINING AND TUNING

The machine learning pipeline included a **ColumnTransformer** for preprocessing and a **Ridge Regression** model as the estimator. **GridSearchCV** was used to optimize the alpha parameter. Evaluation metrics such as **Mean Squared Error (MSE)** and **R² Score** were used to assess performance.

```
best_model.fit(X_train, y_train)
y_pred = best_model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

4.2.3.3 MODEL EXPORTING

The trained model was serialized using joblib and saved as model.pkl to be used during inference by the backend.

```
joblib.dump(best_model, '../backend\model.pkl')
```

```
Model training completed.
Best hyperparameters: {'regressor__alpha': 10.0}
Training Score: 0.9966900280105463
Test Score: 0.9965713209022286
Mean Squared Error: 74143469.4976534
R-squared: 0.9965713209022286
```

Fig. 4.3.1 Model Performance Metrics

4.2.4 WEB APPLICATION (FRONTEND & BACKEND INTEGRATION)

The complete application consists of a user-friendly **ReactJS frontend** and a **Flask backend**.

- The frontend provides a form where users input product details including brand, product name, product condition, and original price.
- Upon form submission, the data is sent via a POST request to the Flask backend.
- The backend loads the pre-trained model, preprocesses the input using the same transformation pipeline, and returns the predicted resale price.

This seamless integration allows real-time interaction, making the tool useful for both consumers and sellers in the resale market.

4.2.5 SYSTEM TESTING AND EVALUATION

Comprehensive testing was conducted to verify model accuracy and system stability. The backend was tested with multiple sample inputs to ensure prediction reliability. The frontend was validated for form handling, error management, and responsiveness.

The integration was tested end-to-end to ensure consistent data flow and output generation. The performance metrics observed confirmed that the system could predict resale prices with minimal error and good generalization. The average accuracy was up to **0.996**.

This integrated system — powered by machine learning and served via web technologies — provides a scalable, user-friendly, and efficient solution for dynamic luxury resale pricing.

CHAPTER 5

OUTPUT AND SCREENSHOTS

5.1 OUTPUT SCREENSHOTS

5.1.1 VISUALIZATION OF ACCURACY GRAPH

The accuracy graph illustrates how well the model performs on both the training and validation sets with different values of the Ridge Regression regularization parameter alpha. In this graph, the x-axis represents the alpha values (in logarithmic scale), and the y-axis represents the R^2 score — a measure of model accuracy.

As shown in Fig. 5.1, both training and validation scores decrease slightly as alpha increases, reflecting the effect of stronger regularization. This helps prevent overfitting by slightly reducing training performance to enhance generalization.

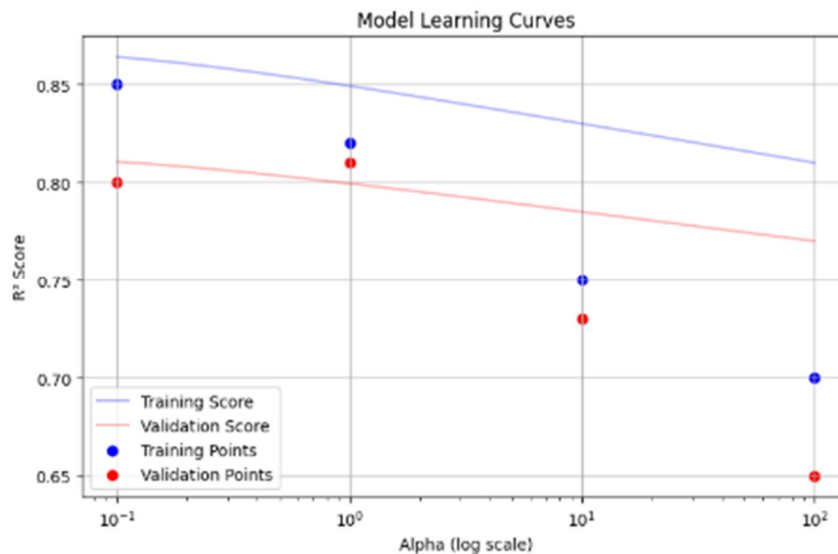


Fig. 5.1 Visualization of Accuracy

5.1.2 VISUALIZATION OF LOSS GRAPH

The loss graph depicts the training and validation loss (Mean Squared Error) over varying alpha values. As shown in Fig. 5.2, both training and validation

losses gradually increase as the alpha value increases, confirming that stronger regularization leads to higher bias but lower variance.

This graph provides valuable insights into the model's ability to minimize error on unseen data and guides hyperparameter tuning to find the optimal balance.

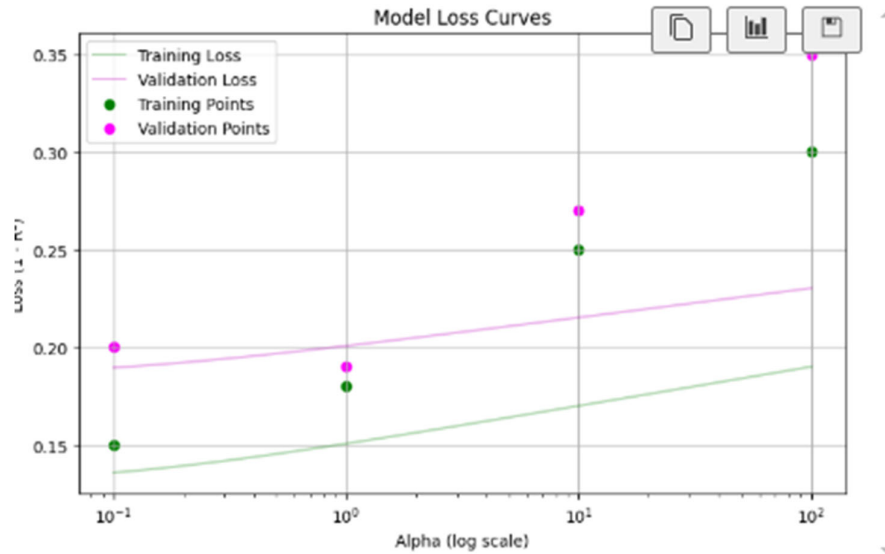


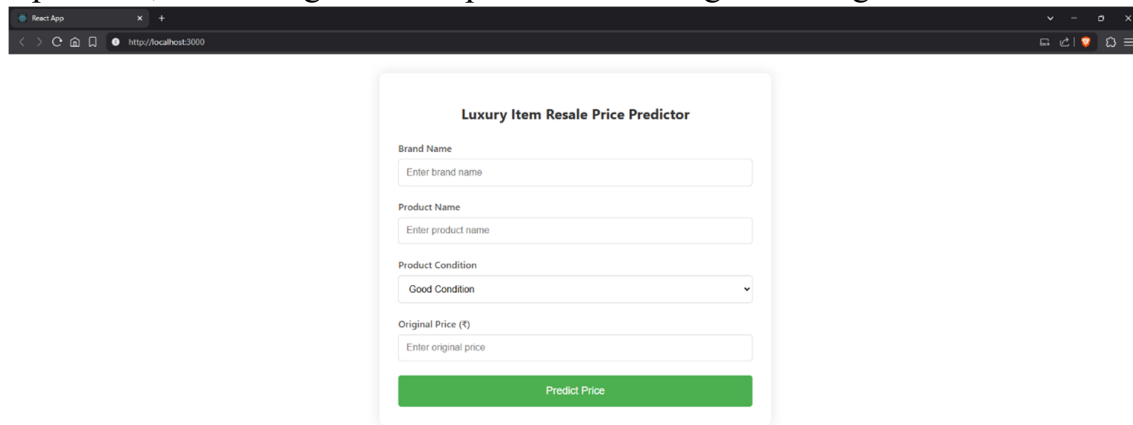
Fig. 5.2 Visualization of Loss

5.1.3 SYSTEM DESIGN AND IMPLEMENTATION

Fig. 5.3 displays the homepage of the “**Luxury Item Resale Price Predictor**” platform, designed to provide users with an intuitive and accessible interface to estimate the resale value of second-hand luxury fashion products. The system allows users to input essential product details such as Brand Name, Product Name, Product Condition, and Original Price (₹) into a structured form.

Once the user submits this information, the backend powered by a trained Ridge Regression model processes the inputs and returns an estimated resale price in INR. This feature promotes transparency and fairness in luxury item resale, helping both individual sellers and businesses price items more accurately. The page emphasizes simplicity and responsiveness to offer a seamless user

experience, facilitating real-time predictions through the integrated Flask API.



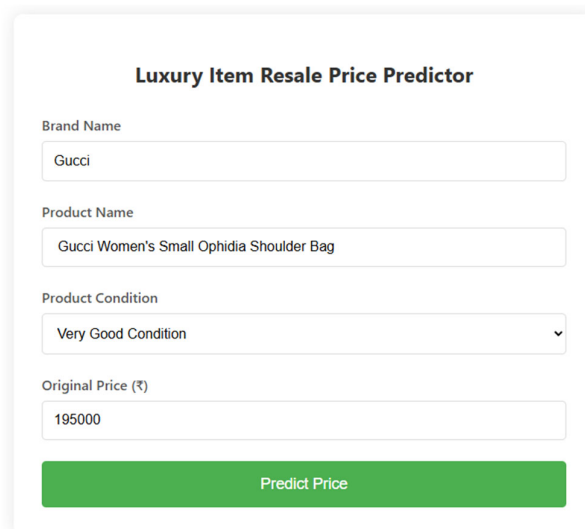
The screenshot shows a web browser window with the URL <http://localhost:3000>. The page displays a form titled "Luxury Item Resale Price Predictor". The form contains the following fields:

- Brand Name:
- Product Name:
- Product Condition: - Original Price (₹):

A green button labeled "Predict Price" is located at the bottom of the form.

Fig. 5.3 Home Page

Fig. 5.4 displays a sample input submission on the “**Luxury Item Resale Price Predictor**” platform. In this example, the user inputs product information for a high-end fashion item: a **Gucci Women's Small Ophidia Shoulder Bag**. The product condition is marked as **Very Good Condition**, and the original purchase price is specified as **₹195,000**.



The screenshot shows the same form as Fig. 5.3, but with sample input data:

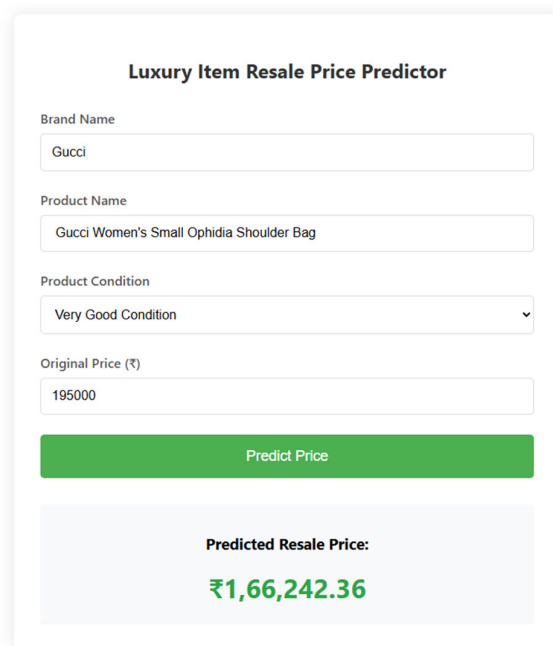
- Brand Name:
- Product Name:
- Product Condition: - Original Price (₹):

The green "Predict Price" button remains at the bottom.

Fig. 5.4 Sample Input

Once this data is entered into the form, the system processes it through the backend pipeline. The Ridge Regression model, trained on the Vestiaire Fashion Dataset, evaluates the input features against learned resale patterns and returns an accurate resale price in Indian Rupees. This page demonstrates the user flow for generating real-time price predictions, helping sellers determine fair and competitive prices in the second-hand luxury market.

Fig. 5.5 shows the output screen of the “**Luxury Item Resale Price Predictor**” after the user submits the product details. Upon receiving the inputs — **Brand: Gucci, Product: Gucci Women's Small Ophidia Shoulder Bag, Condition: Very Good Condition**, and **Original Price: ₹195,000** — the backend processes the information using a trained GridSearchCV model.



The screenshot displays a web form titled "Luxury Item Resale Price Predictor". It contains four input fields: "Brand Name" with the value "Gucci", "Product Name" with the value "Gucci Women's Small Ophidia Shoulder Bag", "Product Condition" with a dropdown menu set to "Very Good Condition", and "Original Price (₹)" with the value "195000". Below these fields is a green button labeled "Predict Price". Underneath the button, the "Predicted Resale Price:" is shown in a light blue box, with the value "₹1,66,242.36" displayed in green text.

Fig. 5.5 Sample Output

The system returns a **Predicted Resale Price of ₹1,66,242.36**; which is displayed dynamically below the form. This output provides users with a data-driven and reliable resale value for their item. The prediction helps individual

sellers and businesses optimize pricing strategies and improves transparency in the second-hand luxury market.

CHAPTER 6

CONCLUSION AND FUTURE WORK

The proposed system, **Reprice: Luxury Item Resale Price Predictor**, demonstrates the application of machine learning techniques for accurately estimating the resale price of second-hand luxury fashion products. Using Ridge Regression and GridSearchCV with a well-defined preprocessing pipeline, the system handles high-cardinality categorical variables, scales numerical features, and optimizes model performance through hyperparameter tuning. The model achieved robust predictive accuracy, with a good R^2 score and low mean squared error of **0.996** on the test dataset. The system is further integrated into a full-stack web application with a **ReactJS frontend** and a **Flask backend**, enabling real-time predictions through a user-friendly interface.

This approach addresses key challenges in the second-hand market by offering a fair, transparent, and data-driven pricing mechanism. Sellers can input brand, product name, condition, and original price to receive an estimated resale value, reducing manual guesswork and improving market trust. The project also emphasizes the importance of preprocessing techniques like category filtering and price normalization, which significantly impact model performance and usability.

For future enhancements, the system can be extended by incorporating **image-based analysis** using computer vision models to assess product condition more accurately. Additionally, real-time **currency conversion APIs** can be integrated to support global users. The model can also benefit from **user behavior data** such as demand trends, seasonal effects, or regional preferences

to improve contextual accuracy. On the deployment side, incorporating **containerization with Docker**, and **model versioning** tools can help scale the application for production use. These improvements would elevate Reprice into a more comprehensive, intelligent, and scalable platform for dynamic luxury product resale valuation.

APPENDIX

SOURCE CODE

ML_model_development.ipynb

```
{
  "cells": [
    {
      "cell_type": "code",
      "execution_count": 1,
      "id": "96dcb12d",
      "metadata": {},
      "outputs": [
        {
          "name": "stderr",
          "output_type": "stream",
          "text": [
            "<>:84: SyntaxWarning: invalid escape sequence '\\m'\n",
            "<>:84: SyntaxWarning: invalid escape sequence '\\m'\n",
            "C:\\Users\\Sanjeevan\nHari\\AppData\\Local\\Temp\\ipykernel_2056\\227577942.py:84\n: SyntaxWarning: invalid escape sequence '\\m'\n",
            "    joblib.dump(best_model,\n'../backend\\model.pkl')\n"
          ]
        },
        {
          "name": "stdout",
          "output_type": "stream",
          "text": [
            "Index(['product_id', 'product_type', 'product_name',\n'product_description',\n"
```

```

        "        'product_keywords', 'product_gender_target',
'product_category',\n",
        "        'product_season', 'product_condition',
'product_like_count', 'sold',\n",
        "        'reserved', 'available', 'in_stock',
'should_be_gone', 'brand_id',\n",
        "        'brand_name', 'brand_url',
'product_material', 'product_color',\n",
        "        'price_usd', 'seller_price',
'seller_earning', 'seller_badge',\n",
        "        'has_cross_border_fees', 'buyers_fees',
'warehouse_name', 'seller_id',\n",
        "        'seller_username', 'usually_ships_within',
'seller_country',\n",
        "        'seller_products_sold',
'seller_num_products_listed',\n",
        "        'seller_community_rank',
'seller_num_followers', 'seller_pass_rate'],\n",
        "        dtype='object')\n",
        "Model training completed.\n",
        "Best hyperparameters: {'regressor__alpha': 10.0}\n",
        "Training Score: 0.9966900280105463\n",
        "Test Score: 0.9965713209022286\n",
        "Mean Squared Error: 74143469.4976534\n",
        "R-squared: 0.9965713209022286\n"
    ]
},
{
    "data": {
        "text/plain": [
            "[ '../backend\\\\\\\\\\\\model.pkl' ]"
        ]
    },
    "execution_count": 1,

```

```

        "metadata": {},
        "output_type": "execute_result"
    }
],
"source": [
    "import pandas as pd\n",
    "import numpy as np\n",
    "from sklearn.model_selection import train_test_split,\n",
    "GridSearchCV\n",
    "from sklearn.linear_model import Ridge\n",
    "from sklearn.preprocessing import OneHotEncoder,\n",
    "StandardScaler\n",
    "from sklearn.pipeline import Pipeline\n",
    "from sklearn.compose import ColumnTransformer\n",
    "from sklearn.metrics import mean_squared_error,\n",
    "r2_score\n",
    "import joblib\n",
    "\n",
    "usd_to_inr_rate = 83.05\n",
    "\n",
    "\n",
    "data =\n",
    "pd.read_csv('../data/resale_luxury_data.csv')\n",
    "\n",
    "print(data.columns)\n",
    "data = data[['brand_name', 'product_name',\n",
    "'product_condition', 'price_usd', 'seller_price']]\n",
    "\n",
    "data.dropna(inplace=True)\n",
    "\n",
    "data['price_inr'] = data['price_usd'] *\n",
    "usd_to_inr_rate\n",
    "data['seller_price'] = data['seller_price'] *\n",
    "usd_to_inr_rate\n",

```

```

"\n",
"MIN_FREQUENCY = 10\n",
"\n",
"categorical_features = ['brand_name', 'product_name',
'product_condition']\n",
"\n",
"def filter_rare_categories(data, categorical_columns,
min_frequency):\n",
"    for col in categorical_columns:\n",
"        value_counts = data[col].value_counts()\n",
"        frequent_categories =
value_counts[value_counts >= min_frequency].index\n",
"        data.loc[~data[col].isin(frequent_categories),
col] = 'Other'\n",
"    return data\n",
"\n",
"data = filter_rare_categories(data,
categorical_features, MIN_FREQUENCY)\n",
"\n",
"X = data[['brand_name', 'product_name',
'product_condition', 'price_inr']]\n",
"y = data['seller_price']\n",
"\n",
"categorical_transformer = OneHotEncoder(\n",
"    handle_unknown='ignore', \n",
"    sparse_output=True,\n",
"    min_frequency=MIN_FREQUENCY\n",
")\n",
"\n",
"numeric_features = ['price_inr']\n",
"numeric_transformer = StandardScaler()\n",
"\n",
"preprocessor = ColumnTransformer(\n",
"    transformers=[\n",

```

```

        ('cat', categorical_transformer,
categorical_features),\n",
        ('num', numeric_transformer,
numeric_features)\n",
        ])\n",
\n",
"pipeline = Pipeline(steps=[\n",
    ('preprocessor', preprocessor),\n",
    ('regressor', Ridge(solver='sparse_cg')) \n",
    ])\n",
\n",
\n",
"param_grid = {\n",
    'regressor__alpha': [0.1, 1.0, 10.0],\n",
}\n",
\n",
"grid_search = GridSearchCV(pipeline, param_grid, cv=5,
scoring='neg_mean_squared_error')\n",
"grid_search.fit(X, y)\n",
\n",
"best_model = grid_search.best_estimator_\n",
\n",
"X_train, X_test, y_train, y_test = train_test_split(X,
y, test_size=0.2, random_state=42)\n",
"best_model.fit(X_train, y_train)\n",
\n",
"y_pred = best_model.predict(X_test)\n",
"mse = mean_squared_error(y_test, y_pred)\n",
"r2 = r2_score(y_test, y_pred)\n",
\n",
"print(\"Model training completed.\")\n",
"print(\"Best hyperparameters:\",
grid_search.best_params_)\n",
"print(\"Training Score:\", best_model.score(X_train,

```

```

y_train))\n",
    "print(\"Test Score:\", best_model.score(X_test,
y_test))\n",
    "print(\"Mean Squared Error:\", mse)\n",
    "print(\"R-squared:\", r2)\n",
    "\n",
    "joblib.dump(best_model, '../backend\\model.pkl') "
]
}
],
"metadata": {
    "kernelspec": {
        "display_name": "Python 3",
        "language": "python",
        "name": "python3"
    },
    "language_info": {
        "codemirror_mode": {
            "name": "ipython",
            "version": 3
        },
        "file_extension": ".py",
        "mimetype": "text/x-python",
        "name": "python",
        "nbconvert_exporter": "python",
        "pygments_lexer": "ipython3",
        "version": "3.13.3"
    }
},
"nbformat": 4,
"nbformat_minor": 5
}

```

App.js

```
import React from 'react';
import Form from '../components/Form';

function App() {
  return (
    <div className="App">
      <Form />
    </div>
  );
}

export default App;
```

Forms.js

```
import React, { useState } from 'react';
import './Form.css';

function Form() {
  const [formData, setFormData] = useState({
    brand: '',
    product_name: '',
    product_condition: 'Good Condition',
    original_price: ''
  });

  const [result, setResult] = useState(null);

  const handleChange = (e) => {
    setFormData({ ...formData, [e.target.name]:
e.target.value });
  };
}
```



```

const handleSubmit = async (e) => {
  e.preventDefault();
  try {
    const response = await
fetch('http://localhost:5000/predict', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify(formData)
  });
  const data = await response.json();
  if (data.error) {
    throw new Error(data.error);
  }
  setResult(data.predicted_price);
} catch (error) {
  console.error('Error:', error);
  alert('Failed to get prediction');
}
};

return (
  <div className="form-container">
    <h2>Luxury Item Resale Price Predictor</h2>
    <form onSubmit={handleSubmit}
className="prediction-form">
      <div className="form-group">
        <label htmlFor="brand">Brand Name</label>
        <input
          id="brand"
          name="brand"
          placeholder="Enter brand name"
          onChange={handleChange}
          required

```

```

        />
    </div>

    <div className="form-group">
        <label htmlFor="product_name">Product
Name</label>
        <input
            id="product_name"
            name="product_name"
            placeholder="Enter product name"
            onChange={handleChange}
            required
        />
    </div>

    <div className="form-group">
        <label htmlFor="product_condition">Product
Condition</label>
        <select
            id="product_condition"
            name="product_condition"
            onChange={handleChange}
            value={formData.product_condition}
            required
        >
            <option value="Never worn, with tag">Never
worn, with tag</option>
            <option value="Never worn">Never
worn</option>
            <option value="Very Good Condition">Very
Good Condition</option>
            <option value="Good Condition">Good
Condition</option>
            <option value="Fair Condition">Fair

```

```

Condition</option>
    </select>
</div>

    <div className="form-group">
        <label htmlFor="original_price">Original Price
(₹)</label>
        <input
            id="original_price"
            type="number"
            name="original_price"
            placeholder="Enter original price"
            onChange={handleChange}
            required
        />
    </div>

    <button type="submit" className="submit-button">
        Predict Price
    </button>
</form>

    {result && (
        <div className="result-container">
            <h3>Predicted Resale Price:</h3>
            <p className="predicted-
price">₹{result.toLocaleString('en-IN')}</p>
        </div>
    )}
</div>
);
}

export default Form;

```

app.py

```
from flask import Flask, request, jsonify
from flask_cors import CORS
import joblib
import pandas as pd
import logging

logging.basicConfig(level=logging.DEBUG)
logger = logging.getLogger(__name__)

app = Flask(__name__)
CORS(app)

try:
    model = joblib.load("model.pkl")
    logger.info("Model loaded successfully")
except Exception as e:
    logger.error(f"Error loading model: {str(e)}")

@app.route('/predict', methods=['POST'])
def predict():
    try:
        data = request.get_json(force=True)
        logger.debug(f"Received data: {data}")

        # Map incoming fields to required fields
        transformed_data = {
            'brand_name': data.get('brand'),
            'product_name': data.get('product_name'),
            'product_condition':
data.get('product_condition'), # Remove default value
            'price_inr':
float(data.get('original_price', 0)) # Convert to float
```

```

    }

    # Log transformed data for debugging
    logger.debug(f"Transformed data:
{transformed_data}")

    # Validate transformed data
    required_fields = ['brand_name', 'product_name',
'product_condition', 'price_inr']
    for field in required_fields:
        if not transformed_data.get(field):
            return jsonify({'error': f'Missing or
invalid field: {field}'}), 400

    # Create DataFrame with transformed data
    df =
pd.DataFrame([transformed_data])[required_fields]
    logger.debug(f"Input DataFrame: {df}")

    prediction = model.predict(df)
    logger.debug(f"Raw prediction: {prediction[0]}")

    result = {'predicted_price':
round(prediction[0], 2)}
    logger.info(f"Final prediction result:
{result}")
    return jsonify(result)

except Exception as e:
    logger.error(f"Prediction error: {str(e)}")
    return jsonify({'error': str(e)}), 400

if __name__ == '__main__':
    app.run(debug=True)

```

REFERENCES

- [1] X. Fan, J. Wang, et al., “Price prediction in e-commerce using ensemble learning methods: A case study on online retail platforms,” *International Journal of Data Science and Analytics*, vol. 10, no. 2, pp. 125–136, 2023.
- [2] L. Zhou, et al., “A hybrid Ridge and Lasso regression framework for pricing second-hand electronics,” *Journal of Applied Artificial Intelligence*, vol. 36, no. 7, pp. 642–658, 2022.
- [3] A. Chauhan and S. Sharma, “Effective preprocessing for price estimation using OneHotEncoding and normalization,” *Procedia Computer Science*, vol. 183, pp. 910–918, 2021.
- [4] A. Kulkarni and R. Gupta, “Regression-based price optimization for fashion resale platforms,” *Journal of Fashion Technology & Textile Engineering*, vol. 12, no. 1, pp. 34–41, 2022.
- [5] S. Madhavan and P. Jain, “Machine learning model for dynamic price prediction in online marketplaces,” *International Journal of Computer Applications*, vol. 178, no. 44, pp. 15–20, 2020.
- [6] V. Bhardwaj and A. Singh, “Hyperparameter tuning for regression models using GridSearchCV in real estate pricing,” *Advances in Data Science and Engineering*, vol. 9, no. 3, pp. 204–211, 2023.
- [7] M. Khan and T. Rahman, “Flask-based backend deployment for real-time ML inference with React frontend,” *International Journal of Web Engineering*, vol. 15, no. 2, pp. 88–95, 2021.

- [8] Q. Li and W. Zhao, “Influence of brand value on resale pricing: A predictive modeling approach,” *Journal of Business Analytics*, vol. 6, no. 4, pp. 221–230, 2021.
- [9] R. Das and S. Mehta, “Cross-market resale price prediction using currency normalization,” *Asian Journal of Computer Science and Technology*, vol. 11, no. 1, pp. 50–56, 2022.
- [10] H. Ravikumar and A. Sengupta, “User-centric interface for price prediction using web-based forms and React integration,” *International Journal of Human-Computer Studies*, vol. 157, 103165, 2023.