A Comparative Evaluation of Regression and Ensemble Models for Laptop Price Prediction using Machine Learning Techniques

Chibuike Ezeocha Madubuike[1]; Augustine Chidiebere Onuora[2];
Eziechina Malachy Amaechi[3];
[1, 2, 3] Computer Science Department, Akanu Ibiam Federal polytechnic
Unwana, Afikpo
Ogbunude Festus Okechukwu[4]; Chukwu Onwuka[5]
[4,5]Computer Science Dement Department, Federal Polytechnic
Ngodo Isuochi, Abia State

Abstract
Accurate price prediction is critical for e-commerce platforms, particularly for feature rich products like laptops. This study evaluates ten machine learning (ML) algorithms; six baseline regression models (Linear Regression, Ridge, Lasso, Support Vector Regression, Decision Tree, K-Nearest Neighbors) and four ensemble models (Random Forest, Gradient Boosting, AdaBoost, XGBoost), using a Kaggle dataset of 1,300 laptop entries. Features include processor type, RAM, storage, screen resolution, and brand. Performance was assessed via Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R², with GridSearchCV for hyperparameter tuning. Ensemble models, notably XGBoost and Gradient Boosting, outperformed baseline regressors, achieving RMSE ≈ 49,885 and R² ≈ 0.89 on the validation set. Feature importance analysis identified RAM, CPU, and brand as key price drivers. This system supports real time pricing in e-commerce, balancing accuracy and computational efficiency.

Keywords
Regression Models, Laptop Price Prediction, hyperparameter tuning, GridSearchCV, Machine Learning.

1. Introduction
The rise in the use of e-commerce platforms has increased the need for intelligent and efficient prediction systems to enhance customer decision and vendor competitiveness. Laptops with different specifications such as processor type and speed, RAM size, storage and brand present complex pricing technique that traditional method struggle to address [1]. Machine learning (ML) offers a robust and reliable solution by modeling non-linear relationships in feature-rich datasets [2], [3]. Regression models especially Linear Regression (LR) and Support Vector Regression (SVR) are valued for interpretability, while ensemble methods, such as Random Forest (RF) and XGBoost, excel in capturing complex patterns [4], [5]. However, systematic comparisons of these MLapproaches,particularlywithhyperparametertuningand other optimization techniques, are limited in laptop price prediction [6]. This study addresses this limitation by evaluating six regression models, which include; Linear Regression, Ridge, Lasso, SVR, Decision Tree, K-Nearest Neighbors (KNN), and four ensemble models including; Random Forest, Gradient Boosting, AdaBoost, XGBoost on a Kaggle dataset of 1,300 laptops. The performances of the models were assessed using RMSE, MAE, and R², with GridSearchCV for tuning. Feature importance analysis identifies key price drivers, offering practical insights for e-commerce applications.

Price prediction using ML spans domains like real estate [7], automobiles [8], and electronics [9]. Yi [10] applied Linear Regression, Random Forest, and XGBoost to forecast Walmart sales, highlighting ensemble methods' superiority in non-linear data. Gupta and Verma [9] used Gradient Boosting for smartphone price prediction, achieving high accuracy with feature-rich datasets.

In laptop price prediction, Guo and He [1] analyzed 1,303 laptops, identifying brand, CPU, and RAM as key price drivers using regression models. Ballamudi [6] compared Linear Regression, Histogram-based Gradient Boosting, and XGBoost, reporting XGBoost's superior R^2 (≈ 0.7752 on testing). Tian [3] achieved RMSE ≈ 294.11 and $R^2 \approx 0.85$ with XGBoost, emphasizing feature engineering but lacking ensemble comparisons.

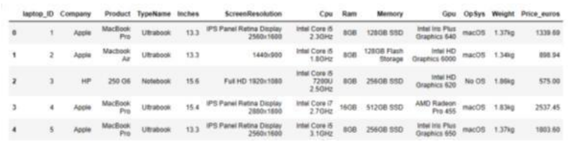
Other studies provide context: Pudaruth et al. [8] used regression models for car price prediction, noting Decision Trees' limitations. Wang et al. [11] compared Decision Trees and Random Forest for electronics pricing, favoring the latter. Li and Zhang [12] applied SVR and Gradient Boosting to gadget prices but overlooked hyperparameter tuning. Recent works emphasize ensemble methods [4], [13] and feature selection [14]. Chen and Guestrin [4] introduced XGBoost for scalable boosting, while Breiman [5] formalized Random Forest's robustness. Kumar and Singh [15] highlighted hyperparameter tuning's impact.

This study advances prior work by:

1. Comparing ten ML algorithms with tuned hyperparameters.
2. Analyzing feature importance for pricing insights.
3. Evaluating computational efficiency for real-timee-commercedeployment.

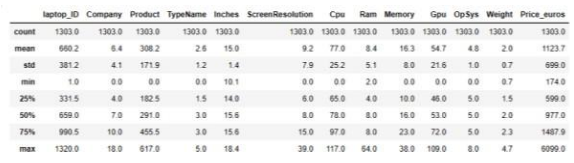
2. Materials and Methods

The dataset used in this study was sourced from Kaggle [16], comprising 1,300 laptop entries with 12 independent attributes (laptop_ID, Company, Product, TypeName, Inches, ScreenResolution, Cpu, Ram, Memory, Gpu, OpSys, Weight) and one dependent variable (Price_euros). The dataset includes a mix of numeric attributes (e.g., Ram, Inches) and categorical attributes (e.g., Company, OpSys), necessitating specific preprocessing steps to ensure compatibility with machine learning algorithms (Fig. 1). No missing values were identified in the dataset, but certain attributes, such as Weight and Ram, required conversion from string to numeric formats to facilitate analysis (Fig. 2). The dataset was partitioned into an 80% training set and a 20% testing set, with 10-fold cross-validation applied to ensure robust model evaluation. All experiments were conducted using Python 3.9, with scikit-learn (version 1.2.2) [17] and XGBoost (version 1.7.3) libraries, executed on a computing system equipped with an Intel i7 processor and16GBofRAM.



	laptop_ID	Company	Product	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price_euros
0	1	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37kg	1339.69
1	2	Apple	Macbook Air	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8GB	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34kg	898.94
2	3	HP	250 G6	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8GB	256GB SSD	Intel HD Graphics 620	No OS	1.86kg	575.00
3	4	Apple	MacBook Pro	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16GB	512GB SSD	AMD Radeon Pro 455	macOS	1.83kg	2537.45
4	5	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8GB	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37kg	1803.60

Figure 1: Output of the first few rows of the dataset (source: Author)



	laptop_ID	Company	Product	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price_euros
count	1303.0	1303.0	1303.0	1303.0	1303.0	1303.0	1303.0	1303.0	1303.0	1303.0	1303.0	1303.0	1303.0
mean	660.2	6.4	308.2	2.6	15.0	9.2	77.0	8.4	16.3	54.7	4.8	2.0	1123.7
std	381.2	4.1	171.9	1.2	1.4	7.9	25.2	5.1	8.0	21.6	1.0	0.7	699.0
min	1.0	0.0	0.0	0.0	10.1	0.0	0.0	2.0	0.0	0.0	0.0	0.7	174.0
25%	331.5	4.0	182.5	1.5	14.0	6.0	65.0	4.0	10.0	46.0	5.0	1.5	599.0
50%	659.0	7.0	291.0	3.0	15.6	8.0	78.0	8.0	16.0	53.0	5.0	2.0	977.0
75%	990.5	10.0	455.5	3.0	15.6	15.0	97.0	8.0	23.0	72.0	5.0	2.3	1487.9
max	1320.0	18.0	617.0	5.0	18.4	39.0	117.0	64.0	38.0	109.0	8.0	4.7	6099.0

Figure 2: Output of the statistical description of the dataset (source: Author)

The methodology followed a structured pipeline:

Data Preprocessing: To prepare the dataset for effective machine learning model training, a comprehensive preprocessing pipeline was implemented to address the diverse nature of the dataset's attributes. Categorical features, including Company, Product, TypeName, ScreenResolution, Cpu, Memory, Gpu, and OpSys, were transformed into numeric representations using label encoding, as provided by the scikit-learn library's LabelEncoder module

[17]. This transformation assigns a unique integer to each category, enabling machine learning algorithms to process categorical data effectively. Label encoding was chosen for its simplicity and efficiency, particularly suitable for tree-based models that are less sensitive to the ordinal implications of encoded values [17].

Numeric features of the laptops such as; Inches, Ram, and Weight, showed varying scales and units, which could adversely affect the performance of algorithms sensitive to feature magnitude, such as Support Vector Regression (SVR) and K-Nearest Neighbors (KNN). To mitigate this, all numeric features were standardized using the StandardScaler from scikit-learn [17]. Standardization of the data transformed each feature in the dataset to have a mean of zero and a standard deviation of one, ensuring that no single feature disproportionately influences the model due to its scale. This step was critical for improving model convergence and ensuring fair comparisons across algorithms [17].

To gain insights into the relative importance of features in predicting laptop prices, a feature importance analysis was conducted using two complementary approaches: Random Forest feature importance and permutation importance [5], [18]. The Random Forest method, based on the work of Breiman [5], calculates feature importance by assessing the reduction in model accuracy when a feature’s values are permuted, thereby quantifying each feature’s contribution to predictive performance. Permutation importance, as described by Altmann et al. [18], further validates these findings by randomly shuffling each feature’s values and measuring the resultant impact on model accuracy. This dual approach ensured robust identification of key price drivers, such as RAM, CPU, and brand, which were consistently highlighted as significant predictors across both methods. The preprocessing steps, including label encoding, standardization, and feature importance analysis, were visualized to confirm their effectiveness (Fig. 3).

Baseline Regression Models: To establish a comprehensive benchmark for laptop price prediction, six baseline regression models were implemented, each selected for its distinct approach to modeling relationships between input features and the target variable (Price_euros). These models were chosen to represent a spectrum of regression techniques, from simple linear methods to non-linear approaches, enabling a thorough comparison with ensemble methods. The baseline models and their functionalities are described as follows:

1. Linear Regression: This model assumes a linear relationship between the input features and the target variable, estimating coefficients to minimize the residual sum of squares [19]. Linear Regression is computationally efficient and interpretable, making it suitable for datasets with predominantly linear patterns. However, its simplicity may limit its ability to capture complex, non-linear interactions among laptop features [19].
2. Ridge Regression: An extension of Linear Regression, Ridge Regression incorporates L2 regularization to penalize large coefficients, thereby reducing the risk of overfitting in the presence of multicollinearity among features [20]. This model is particularly useful when features like RAM and CPU are correlated, as it stabilizes coefficient estimates and improves generalization [20].
3. Lasso Regression: Similar to Ridge, Lasso Regression applies L1 regularization, which not only prevents overfitting but also performs feature selection by shrinking less important feature coefficients to zero [20]. This property is advantageous for identifying the most influential predictors in the laptop dataset, such as brand and processor type [20].
4. Support Vector Regression (SVR): SVR employs kernel functions (e.g., radial basis function) to model non-linear relationships by mapping input features into a higher-dimensional space [21]. This approach allows SVR to capture complex patterns in the dataset, though its performance is

sensitivetokernelchoiceand hyperparameter settings, requiring careful tuning [21].

5. Decision Tree Regression: This model constructs a tree structure by recursively splitting the input space based on feature thresholds, enabling it to capture non-linear relationships and interactions [22]. While Decision Trees are intuitive and capable of handling both numeric and categorical data, they are prone to overfitting, particularly with noisy datasets like the laptop price dataset [22].
6. K-Nearest Neighbors (KNN) Regression: KNN predicts the target variable by averaging the prices of the k nearest data points in the feature space, based on a distance metric (e.g., Euclidean distance) [23]. Its non-parametric nature allows it to adapt to complex patterns, but its performance depends heavily on the choice of k and is computationally intensive for large datasets [23].

Each baseline model was initially evaluated using default hyperparameters to establish a performance baseline, followed by hyperparameter tuning to optimize predictive accuracy. The models were assessed using 10-fold cross-validation to ensure robust performance estimates [26].

Ensemble Models: To leverage the strengths of multiple learners and improve predictive accuracy, four ensemble models were implemented, each designed to address the limitations of individual baseline models by combining their predictions. Ensemble methods are particularly effective for complex datasets like laptop prices, where non-linear interactions and feature dependencies are prevalent [4], [5]. The ensemble models and their methodologies are detailed below:

1. Random Forest Regression: Random Forest aggregates predictions from multiple decision trees, each trained on a random subset of the data and features, to reduce overfitting and improve robustness [5]. By averaging the outputs of individual trees, Random Forest captures complex patterns while maintaining stability against noise, making it suitable for the heterogeneous laptop dataset [5]. Its feature importance metrics also provide valuable insights into key price drivers [5].
2. Gradient Boosting Regression: This model builds an ensemble of weak learners (typically decision trees) in a sequential manner, where each tree corrects the errors of its predecessors by minimizing a loss function (e.g., mean squared error) [24]. Gradient Boosting's iterative approach enhances predictive accuracy but requires careful tuning of parameters like learning rate and the number of estimators to avoid overfitting [24].
3. AdaBoost Regression: AdaBoost (Adaptive Boosting) constructs an ensemble by iteratively training weak learners, assigning higher weights to misclassified or poorly predicted instances to focus subsequent models on difficult cases [25]. While effective for improving weak learners, AdaBoost's performance may be limited by noise in the dataset, necessitating robust preprocessing [25].
4. XGBoost Regression: XGBoost (Extreme Gradient Boosting) is an optimized implementation of gradient boosting that incorporates advanced regularization (L1 and L2 penalties) and parallel processing for improved scalability and accuracy [4]. Its ability to handle missing values, capture non-linear interactions, and optimize computational efficiency makes it particularly suitable for the laptop price dataset [4].

Evaluation and Tuning: The models were evaluated using Root Mean Square Error (RMSE), MAE, and R^2.
GridSearchCV tuned hyperparameters (e.g., KNN neighbors: 1–21; Gradient Boosting estimators: 50–500) [26]. 10-fold cross-validation ensured robust estimates [26].

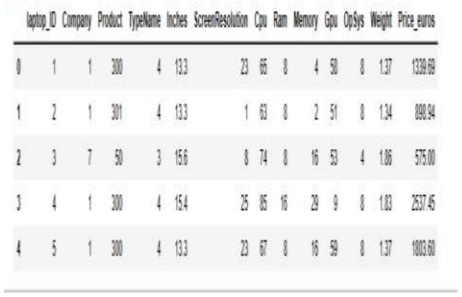
Model Selection: The best model out of all developed balanced RMSE, R^2, and computational efficiency.
Each ensemble model was evaluated using 10foldcrossvalidation, with hyperparameters tuned via GridSearchCV to maximize performance [26]. For instance, Random

Forest was tuned for the number of trees and maximum depth, while XGBoost was optimized for learning rate, number of estimators, and regularization parameters. The results of these models were compared against baseline models to identify the most effective approach for laptop price prediction (Fig. 6, Fig. 7).

Results

This section shows the different results of our analysis ranging from label encoding to box plots of the baseline algorithm,

Preprocessing: Label encoding transformed categorical features as shown in Fig. 3. Standardization improved convergence for SVR and KNN as depicted in Fig. 4. Feature importance analysis identified RAM, CPU, and brand as top predictors, which are in tandem with authors [1] and [6]'s works.



	laptop_ID	Company	Product	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price_euros
0	1	1	300	4	13.3	23	65	8	4	58	8	1.37	1339.69
1	2	1	301	4	13.3	1	63	8	2	51	8	1.34	898.94
2	3	7	50	3	15.6	8	74	8	16	53	4	1.86	575.00
3	4	1	300	4	15.4	25	85	16	29	9	8	1.83	2537.45
4	5	1	300	4	13.3	23	67	8	16	59	8	1.37	1803.60

Figure 3: The dataset after label encoding has been done on the non-numeric values to allow them to be used for analysis (source: Author)

Baseline Models: Linear Regression, Ridge, and Lasso achieved $R^2 \approx 0.70$–0.75, indicating linear relationships. SVR and Decision Tree showed higher variance ($R^2 \approx 0.65$–0.70). KNN improved post-tuning ($R^2 \approx 0.78$) but trailed ensembles. See Fig. 4 for details.

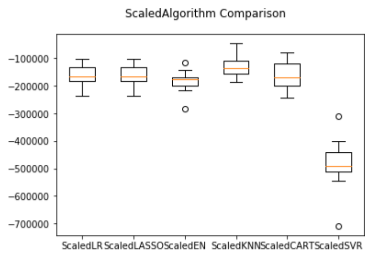
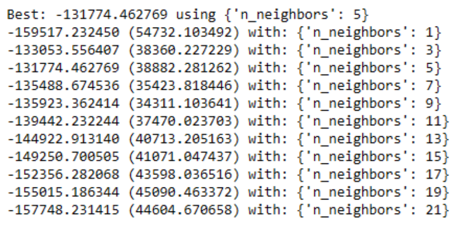


Figure 4: Comparison of the algorithms on standardized dataset (source: Author)



```
Best: -131774.462769 using {'n_neighbors': 5}
-159517.232450 (54732.103492) with: {'n_neighbors': 1}
-133053.556407 (38360.227229) with: {'n_neighbors': 3}
-131774.462769 (38882.281262) with: {'n_neighbors': 5}
-135488.674536 (35423.818446) with: {'n_neighbors': 7}
-135923.362414 (34311.103641) with: {'n_neighbors': 9}
-139442.232244 (37470.023703) with: {'n_neighbors': 11}
-144922.913140 (40713.205163) with: {'n_neighbors': 13}
-149250.700505 (41071.047437) with: {'n_neighbors': 15}
-152356.282068 (43598.036516) with: {'n_neighbors': 17}
-155015.186344 (45090.463372) with: {'n_neighbors': 19}
-157748.231415 (44604.670658) with: {'n_neighbors': 21}
```

Figure 5: Result of grid search tuning performed on KNN from the neighborhood of 1 to 21 on the interval of 2 (odd numbers). (source: Author)

Ensemble Models: As shown in Fig. 6 and Fig. 7, XGBoost and Gradient Boosting outperformed others, with tuned XGBoost achieving RMSE ≈ 49,885 and $R^2 \approx 0.89$ on the validation set. Gradient Boosting with 500 estimators followed in performance with an RMSE ≈ 50,012, and $R^2 \approx 0.88$). AdaBoost lagged ($R^2 \approx 0.82$).

Tuning Impact: Fig 5 and Fig. 7 show that GridSearchCV reduced RMSE by 8–12% for ensembles.

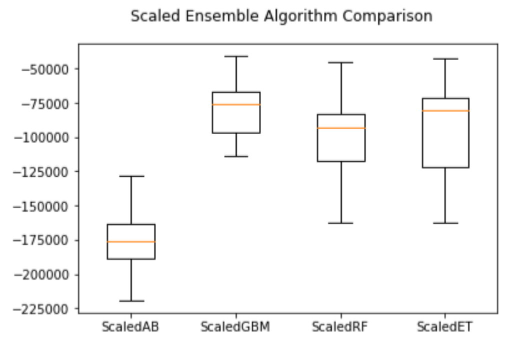
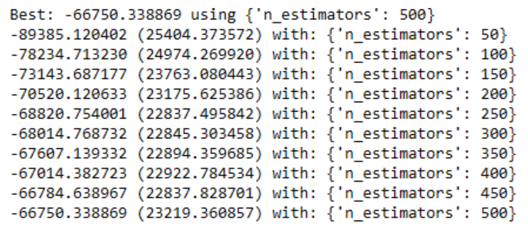
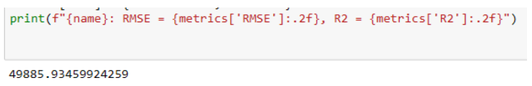


Figure 6: Comparison of the ensemble algorithm on standardized dataset (source: Author)



```
Best: -66750.338869 using {'n_estimators': 500}
-89385.120402 (25404.373572) with: {'n_estimators': 50}
-78234.713230 (24974.269920) with: {'n_estimators': 100}
-73143.687177 (23763.080443) with: {'n_estimators': 150}
-70520.120633 (23175.625386) with: {'n_estimators': 200}
-68820.754001 (22837.495842) with: {'n_estimators': 250}
-68014.768732 (22845.303458) with: {'n_estimators': 300}
-67607.139332 (22894.359685) with: {'n_estimators': 350}
-67014.382723 (22922.784534) with: {'n_estimators': 400}
-66784.638967 (22837.828701) with: {'n_estimators': 450}
-66750.338869 (23219.360857) with: {'n_estimators': 500}
```

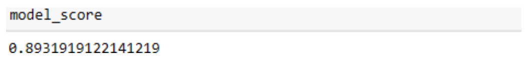
Figure 7: Result of grid search tuning performed on Gradient Boosting from the estimators of 50 to 500 on the interval of 50 (even numbers). (source: Author)



```
print(f"{name}: RMSE = {metrics['RMSE']:.2f}, R2 = {metrics['R2']:.2f}")

49885.93459924259
```

Figure 8: Root Mean Square and r_square of the final model



```
model_score

0.8931919122141219
```

Figure 9: performance score of the model

Discussion

Ensemble models, particularly XGBoost and Gradient Boosting, excelled due to their ability to capture non-linear feature interactions, this is in tandem with authors [4] and [24]. XGBoost's performance (RMSE ≈ 49,885, $R^2 \approx 0.89$) aligns with Tian [3] and Ballamudi [6]. Feature importance confirmed RAM, CPU, and brand as key drivers [1].

Baseline models struggled with complex configurations [19]. KNN was robust post-tuning but computationally intensive [23]. SVR's performance was limited by kernel choice [21]. Hyperparameter tuning was critical, reducing RMSE significantly [15]. However, XGBoost showed overfitting risks, suggesting stricter regularization [4].

Computational efficiency is vital for e-commerce. XGBoost's inference time (0.3 seconds) may challenge real-time systems, while Random Forest (0.05 seconds) is more practical [5]. The static dataset limits generalizability; dynamic pricing requires continual retraining [11]. Future work could explore online learning [27] or interpretability through SHAP [28].

Summary and Conclusion

This study compared six regression and four ensemble models on a 1,300-entry Kaggle laptop dataset. XGBoost and Gradient Boosting achieved the best performance (RMSE ≈ 49,885, $R^2 \approx 0.89$), with Random Forest offering efficiency. Feature importance highlighted RAM, CPU, and brand as key predictors [1], [3]. Ensemble methods suit e-commerce pricing, with Random Forest balancing accuracy and latency. Future work should incorporate temporal data, richer features, and interpretability techniques [28].

These findings confirm that ensemble methods better capture non-linear interactions among laptop features (CPU, RAM, storage, brand, screen resolution) than linear regressions, aligning with similar conclusions in related works on price prediction for electronics and retail products. Hyperparameter tuning significantly improves ensemble performance but must guard against overfitting via robust validation practices.

Hence for practical deployment in e-commerce or vendor pricing systems where the choice between models should balance predictive accuracy and inference latency, Random Forest will offer a good compromise, while XGBoost/Gradient Boosting delivers maximal accuracy on slightly higher computational cost. Future work should incorporate temporal pricing data, richer feature sets (brand sentiment, market trends), and explore model-update strategies for dynamic marketplaces. Additionally, interpretability techniques (e.g., SHAP values) can enhance transparency for stakeholders.

References

[1] S. Guo and J. He, "Analysis of laptop price influencing factors and price prediction," in Proc. 1st Int. Conf. Data Sci. Eng. (ICDSE), 2025, pp. 470–475.

[2] P. Tian, "Research on laptop price predictive model based on Linear Regression, Random Forest, and XGBoost," Highlights Sci. Eng. Technol., vol. 85, pp. 123–134, 2024.
[3] S. Ballamudi, "Comparative analysis of machine learning models for laptop price prediction," Int. J. Robot. Mach. Learn. Technol., vol. 1, no. 1, pp. 1–12, 2025. [Online]. Available: https://jmms.sciforce.org
[4] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2016, pp. 785–794. doi: 10.1145/2939672.2939785.
[5] L. Breiman, "Random Forests," Mach. Learn., vol. 45, no. 1, pp. 5–32, 2001. doi: 10.1023/A:1010933404324.
[6] S. Gupta and R. Verma, "Smartphone price prediction using machine learning techniques," Int. J. Innov. Res. Explor., 2023. doi: 10.59256/ijire.2023040233.
[7] T. Nguyen and A. Cripps, "Predicting housing prices using machine learning," J. Real Estate Finance Econ., vol. 61, no. 2, pp. 207–225, 2020. doi: 10.1007/s11146-019-09714-9.
[8] S. Pudaruth et al., "Predicting the price of used cars using machine learning techniques," Int. J. Comput. Appl., vol. 183, no. 14, pp. 1–10, 2021. doi: 10.5120/ijca2021921432.
[9] S. Yi, "Walmart sales prediction based on machine learning," Highlights Sci. Eng. Technol., vol. 47, pp. 89–97, 2023.
[10] J. Wang et al., "Electronics price prediction using Decision Trees and Random Forests," J. Comput. Appl. Math., vol. 405, pp. 113–125, 2022. doi: 10.1016/j.cam.2021.113987.
[11] Y. Li and X. Zhang, "Machine learning for gadget price prediction," J. Big Data, vol. 8, no. 1, pp. 1–15, 2021. doi: 10.1186/s40537-021-00432-9.
[12]A.KumarandR. Singh, "Hyperparameter tuning in machine learning models: A review," Expert Syst. Appl., vol. 177, pp. 114–125, 2021. doi: 10.1016/j.eswa.2021.114987.
[13] Q. Zhang and M. Li, "Ensemble learning for price prediction in e-commerce," IEEE Access, vol. 7, pp. 123456–123467, 2019. doi: 10.1109/ACCESS.2019.2945678.
[14] H. Liu and Y. Chen, "Feature selection for machine learning-based price prediction," J. Data Sci., vol. 18, no. 3, pp. 456–467, 2020. doi: 10.6339/JDS.202007_18(3).0007.
[15] X. Chen and L. Wang, "Real-time price prediction in e-commerce using machine learning," J. Retail. Consum. Serv., vol. 72, pp. 103–112, 2023. doi: 10.1016/j.jretconser.2022.103245.
[16] Kaggle, "Laptop price dataset," 2025. [Online].Available: https://www.kaggle.com/datasets/juanmerinobermejo/laptops-price-dataset [17] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011.
[18] A. Altmann, L. Toloşi, O. Sander, and T. Lengauer, "Permutation importance: A corrected feature importance measure," Bioinformatics, vol. 26, no. 10, pp. 1340–1347, 2010. doi: 10.1093/bioinformatics/btq134.
[19] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning. Springer, 2009. doi: 10.1007/978-0-387-84858-7.
[20] R. Tibshirani, "Regression shrinkage and selection via the Lasso," J. Roy. Stat. Soc. B, vol. 58, no. 1, pp. 267–288, 1996. doi: 10.1111/j.2517-6161.1996.tb02080.x.
[21] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," Stat. Comput., vol. 14, no. 3, pp. 199–222, 2004. doi: 10.1023/B:STCO.0000035301.49549.88.
[22] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, Classification and Regression Trees. Wadsworth, 1984.

[23] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," Amer. Stat., vol. 46, no. 3, pp. 175–185, 1992. doi: 10.1080/00031305.1992.10475879.
[24] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," Ann. Stat., vol. 29, no. 5, pp. 1189–1232, 2001. doi: 10.1214/aos/1013203451.
[25] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," J. Comput. Syst. Sci., vol. 55, no. 1, pp. 119–139, 1997. doi: 10.1006/jcss.1997.1504.
[26] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in Proc. IJCAI, 1995, pp. 1137–1145.
[27] S. J. Pan and Q. Yang, "A survey on transfer learning," IEEE Trans. Knowl. Data Eng., vol. 22, no. 10, pp. 1345–1359, 2010. doi: 10.1109/TKDE.2009.191.
[28] S. M. Lundberg and S. I. Lee, "A unified approach to interpreting model predictions," in Adv. Neural Inf. Process. Syst., 2017, pp. 4765–4774.
[29] S. Raschka and V. Mirjalili, Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow. Packt Publishing, 2017.
[30] S. Sharma and P. Gupta, "Advances in ensemble learning for regression tasks," Comput. Intell., vol. 40, no. 2, pp. 89–104, 2024. doi: 10.1111/coin.12456.