

Smartphone Price Patterns Prediction Using Machine Learning Algorithms

*A Project Report submitted in the partial fulfillment of the Requirements
for the award of the degree.*

BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING

Submitted by

Y. Rambabu	(21471A05B6)
P. Rajasekhar	(21471A05B5)
B. Suhas	(21471A0578)

Under the Esteemed Guidance of

Mr. Shaik Rafi, M.Tech., (Ph.D)

Asst. Professor



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**NARASARAOPETA ENGINEERING COLLEGE: NARASAROPET
(AUTONOMOUS)**

Accredited by NAAC with A+ Grade and NBA under Tyre -1 NIRF
rank in the band of 201-300 and an ISO 9001:2015 Certified

Approved by AICTE, New Delhi, Permanently Affiliated to JNTUK, Kakinada
KOTAPPAKONDA ROAD, YALAMANDA VILLAGE, NARASARAOPET-
522601

2024-2025

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



CERTIFICATE

This is to certify that the project that is entitled with the name “**Smartphone Price Patterns Prediction Using Machine Learning Algorithms**” is a bonafide work done by the team **Y. Rambabu (21471A05B6), P. Rajasekhar (21471A05B5), B. Suhas(21471A0578)** in partial fulfillment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in the Department of COMPUTER SCIENCE AND ENGINEERING during 2024-2025.

PROJECT GUIDE

Shaik Rafi, M.Tech.,(Ph.D)
Asst.Professor

PROJECT CO-ORDINATOR

Dr. M. Sireesha Moturi, M.Tech.,Ph.D
Assoc. Professor

HEAD OF THE DEPARTMENT

Dr. S. N. Tirumala Rao, M.Tech.,Ph.D
Professor & HOD

EXTERNAL EXAMINER

DECLARATION

We declare that this project work titled "**Smartphone Price Patterns Prediction Using Machine Learning Algorithms**" is composed by ourselves that the work contain here is our own except where explicitly stated otherwise in the text and that this work has not been submitted for any other degree or professional qualification except as specified.

By

Y.Rambabu (21471A05B6)

P.Rajasekhar (21471A05B5)

B.Suhas (21471A0578)

ACKNOWLEDGEMENT

We wish to express my thanks to carious personalities who are responsible for the completion of the project. We are extremely thankful to our beloved chairman sri **M. V. Koteswara Rao, B.Sc.**, who took keen interest in us in every effort throughout thiscourse. We owe out sincere gratitude to our beloved principal **Dr. S. Venkateswarlu, Ph.D.**, for showing his kind attention and valuable guidance throughout the course.

We express our deep felt gratitude towards **Dr. S. N. Tirumala Rao**, M.Tech., Ph.D., HOD of CSE department and also to our guide **Shaik Rafi**, M.Tech.,(Ph.D) **Assistant Professor** of CSE department whose valuable guidance and unstinting encouragement enable us to accomplish our project successfully in time.

We extend our sincere thanks towards **Dr. Sireesha Moturi**, B.Tech, M.Tech.,Ph.D., Associate professor & Project coordinator of the project for extending her encouragement. Their profound knowledge and willingness have been a constant source of inspiration for us throughout this project work.

We extend our sincere thanks to all other teaching and non-teaching staff to department for their cooperation and encouragement during our B.Tech degree.

We have no words to acknowledge the warm affection, constant inspiration and encouragement that we received from our parents.

We affectionately acknowledge the encouragement received from our friends and those who involved in giving valuable suggestions had clarifying out doubts which had really helped us in successfully completing our project.

By

Y.Rambabu (21471A05B6)

P.Rajasekhar (21471A05B5)

B. Suhas (21471A0578)



INSTITUTE VISION AND MISSION

INSTITUTION VISION

To emerge as a Centre of excellence in technical education with a blend of effective student centric teaching learning practices as well as research for the transformation of lives and community,

INSTITUTION MISSION

M1: Provide the best class infra-structure to explore the field of engineering and research

M2: Build a passionate and a determined team of faculty with student centric teaching, imbining experiential, innovative skills

M3: Imbibe lifelong learning skills, entrepreneurial skills and ethical values in students for addressing societal problems



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION OF THE DEPARTMENT

To become a center of excellence in nurturing the quality Computer Science & Engineering professionals embedded with software knowledge, aptitude for research and ethical values to cater to the needs of industry and society.

MISSION OF THE DEPARTMENT

The department of Computer Science and Engineering is committed to

M1: Mould the students to become Software Professionals, Researchers and Entrepreneurs by providing advanced laboratories.

M2: Impart high quality professional training to get expertise in modern software tools and technologies to cater to the real time requirements of the industry.

M3: Inculcate team work and lifelong learning among students with a sense of societal and ethical responsibilities.



Program Specific Outcomes (PSO's)

PSO1: Apply mathematical and scientific skills in numerous areas of Computer Science and Engineering to design and develop software-based systems.

PSO2: Acquaint module knowledge on emerging trends of the modern era in Computer Science and Engineering.

PSO3: Promote novel applications that meet the needs of entrepreneur, environmental and social issues.



Program Educational Objectives (PEO's)

The graduates of the programme are able to:

PEO1: Apply the knowledge of Mathematics, Science and Engineering fundamentals to identify and solve Computer Science and Engineering problems.

PEO2: Use various software tools and technologies to solve problems related to academia, industry and society.

PEO3: Work with ethical and moral values in the multi-disciplinary teams and can communicate effectively among team members with continuous learning.

PEO4: Pursue higher studies and develop their career in software industry.

Program Outcomes

- 1. Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- 2. Problem analysis:** Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
- 3. Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
- 4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
- 5. Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
- 6. The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
- 7. Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

- 8. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- 9. Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- 10. Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
- 11. Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
- 12. Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Project Course Outcomes (CO'S):

CO421.1: Analyze the System of Examinations and identify the problem.

CO421.2: Identify and classify the requirements.**CO421.3:** Review the Related Literature

CO421.4: Design and Modularize the project

CO421.5: Construct, Integrate, Test and Implement the Project.

CO421.6: Prepare the project Documentation and present the Report using appropriate method.

Course Outcomes – Program Outcomes mapping

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
C421.1		✓											✓		
C421.2	✓		✓		✓								✓		
C421.3				✓		✓	✓	✓					✓		
C421.4			✓			✓	✓	✓					✓	✓	
C421.5					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
C421.6									✓	✓	✓		✓	✓	

Course Outcomes – Program Outcome correlation

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
C421.1	2	3											2		
C421.2			2		3								2		
C421.3				2		2	3	3					2		
C421.4			2			1	1	2					3	2	
C421.5					3	3	3	2	3	2	2	1	3	2	1
C421.6									3	2	1		2	3	

Note: The values in the above table represent the level of correlation between CO's and PO's:

1. Low level
2. Medium level
3. High level

Project mapping with various courses of Curriculum with Attained PO's:

Name of the course from which principles are applied in this project	Description of the device	Attained PO
C2204.2, C22L3.2	Gathering the requirements and defining the problem, plan to develop a model for recognizing image manipulations using CNN and ELA	PO1, PO3
CC421.1, C2204.3, C22L3.2	Each and every requirement is critically analyzed, the process mode is identified	PO2, PO3
CC421.2, C2204.2, C22L3.3	Logical design is done by using the unified modelling language which involves individual team work	PO3, PO5, PO9
CC421.3, C2204.3, C22L3.2	Each and every module is tested, integrated, and evaluated in our project	PO1, PO5
CC421.4, C2204.4, C22L3.2	Documentation is done by all our four members in the form of a group	PO10
CC421.5, C2204.2, C22L3.3	Each and every phase of the work in group is presented periodically	PO10, PO11
C2202.2, C2203.3, C1206.3, C3204.3, C4110.2	Implementation is done and the project will be handled by the social media users and in future updates in our project can be done based on detection of forged videos	PO4, PO7
C32SC4.3	The physical design includes website to check whether an image is real or fake	PO5, PO6

ABSTRACT

Selecting the best smartphone can be challenging due to the wide range of models available on the market. This study shows how the machine learning models can predict mobile phone prices based on their features. We evaluated several machine learning techniques, including Logistic Regression, Decision Trees, Random Forest, SVC, K-Neighbors Classifier, Gaussian Naive Bayes (GaussianNB), AdaBoost, Gradient Boosting, Extra Trees, Bagging Classifiers, and XGBoost.

The primary objective was to identify the most effective model for price forecasting and to investigate the factors influencing phone prices. Our research offers insights to both consumers and manufacturers, helping them make more informed decisions about phone features and pricing. We emphasize the importance of using diverse datasets that accurately represent various smartphone models and pricing points.

Key factors affecting phone costs were identified, and model performance was assessed using metrics such as accuracy, F1-score, and classification reports. Model performance was further enhanced through hyperparameter tuning with GridSearchCV, achieving 97% accuracy with the Decision Tree, K-Neighbors Classifier, SVC, AdaBoost, and Random Forest models.

Among these, the Decision Tree and SVC was selected as the optimal models, offering a good tradeoff between accuracy, flexibility, and time complexity. This study aims to provide valuable data to guide consumers in making informed choices about mobile phone features.

INDEX

1. Introduction.....	16
1.1 machine learning applications.....	19
1.2 Importance of Machine Learning in Smartphone Price Patterns Prediction.....	19
1.3 Importance of Machine Learning using Python.....	20
2. Literature Survey.....	21
2.1 Machine learning.....	21
2.2 Some machine learning methods.....	22
2.3 Machine learning model Architectures.....	23
2.4 Training and Optimization Methods.....	24
2.5 Evaluation and Performance Metrics.....	24
3. Existing System.....	25
4. Proposed System.....	27
5. System Requirements.....	30
5.1 Hardware Requirements.....	30
5.2 Software Requirements.....	30
6. System Analysis.....	31
6.1 Scope of the Project.....	31
6.2 Analysis.....	32
6.3 Data Preprocessing.....	33
6.4 Feature Selection and Optimization.....	34
6.5 Model Building and Training.....	36
6.6 Comparative Analysis.....	39
7. System Design.....	44
8. Implementation.....	46
9. Result Analysis.....	80
10. Test Cases	90
11. User Interface.....	91
12. Conclusion.....	93
13. Future Scope.....	94
14. References.....	96

LIST OF FIGURES

1. Fig 4.1 Flow chart of proposed system.....	28
2. Fig 6.4 Chi-Square Test.....	35
3. Fig 6.5 Lasso method.....	35
4. Fig 7.1 Design Overview.....	43
5. Fig 9.1 Model Accuracies.....	79
6. Fig 9.2 Co-relation matrix.....	80
7. Fig 9.3 Confusion matrix of decision tree.....	81
8. Fig 9.4 Confusion matrix of SVC.....	81
9. Fig 9.5 ROC Curve of SVC.....	82
10. Fig 9.6 ROC Curve of decision tree.....	82
11. Fig 10.1 Prediction as Low cost.....	84
12. Fig 10.2 Prediction as medium cost	84
13. Fig 10.3 Prediction as high cost.....	84
14. Fig 10.4 Prediction as very high cost.....	84
15. Fig 11.1 Home screen	85
16. Fig 11.2 prediction screen	85
17. Fig 11.3 Contact-us screen	86
18. Fig 11.4 About-us screen	86

1. INTRODUCTION

With the rapid advancements in technology, smartphones have become an essential part of daily life. Over the past decade, the smartphone industry has experienced exponential growth, driven by innovations in hardware, software, and connectivity. Smartphones are no longer just communication devices—they serve as multifunctional tools that support social networking, internet browsing, multimedia consumption, gaming, health monitoring, navigation, and more. As a result, the smartphone market has become highly competitive, with manufacturers introducing new models at a rapid pace to meet consumer demands. Consumers are now faced with a vast range of smartphone models, each with varying specifications, features, and price points. Choosing the right smartphone has become a daunting task due to the increasing number of available options and the rapid turnover of models. The complex interplay of technology, brand perception, and market dynamics makes it difficult for consumers to assess the true value of a smartphone. Consequently, understanding the factors that influence smartphone pricing has become a critical area of research for both consumers and manufacturers.

The pricing of smartphones is influenced by various factors, including hardware specifications, brand value, market trends, and emerging technologies [1]. Key hardware specifications such as processor speed, RAM capacity, battery life, screen resolution, storage capacity, and camera quality play a significant role in determining the price of a smartphone. Higher-end smartphones with powerful processors, advanced graphics capabilities, and premium build materials command higher prices, while budget smartphones are designed to provide basic functionality at lower costs. Brand value and market positioning also significantly impact smartphone pricing. Established brands such as Apple and Samsung are able to command premium prices due to their brand reputation and customer loyalty, while newer or lesser-known brands often compete on price to gain market share. Additionally, market trends such as the rise of 5G connectivity, foldable displays, and enhanced artificial intelligence (AI) capabilities influence the pricing of new smartphone models. Emerging technologies like under-display fingerprint sensors, fast charging, and enhanced camera algorithms also contribute to price variations [2].

Machine learning (ML) has emerged as a powerful tool in predictive analysis, providing a systematic approach to solving complex problems across various industries. ML models have shown remarkable success in fields such as healthcare, finance, marketing, and manufacturing. In the context of smartphone pricing, ML algorithms can analyze large datasets of historical smartphone data to identify patterns and relationships between different features and price points. By training ML models on datasets that include smartphone attributes such as RAM, battery capacity, processor speed, screen resolution, storage, and network compatibility, accurate price predictions can be generated. Predicting smartphone prices using ML models can help consumers make informed purchasing decisions and enable manufacturers to optimize pricing strategies. For consumers, an accurate price prediction model can provide insights into whether a smartphone is priced fairly based on its features and market positioning. For manufacturers and retailers, ML-driven insights can aid in competitive pricing, inventory management, and market segmentation [3].

Several machine learning models have been applied to smartphone price prediction, each with its strengths and limitations. Logistic Regression, a widely used classification algorithm, is effective in handling binary classification problems and provides interpretable outputs. However, its performance may be limited when dealing with complex nonlinear relationships. Decision Trees, on the other hand, offer a more flexible approach by creating a tree-like structure that partitions the data based on feature values. While Decision Trees are easy to interpret, they are prone to overfitting, especially with small datasets. Random Forest, an ensemble learning technique, addresses the limitations of Decision Trees by creating multiple decision trees and aggregating their outputs to improve accuracy and reduce overfitting. Support Vector Classifier (SVC) is particularly effective in handling high-dimensional data and separating complex data points using hyperplanes. However, SVC can be computationally expensive with large datasets. The K-Neighbors Classifier (KNN) is a simple but effective algorithm that classifies data points based on their proximity to neighboring data points. While KNN works well with small datasets, it becomes inefficient with large datasets due to the computational cost of distance calculations [4]. To enhance model performance, advanced techniques such as hyperparameter tuning and feature selection are employed. Hyperparameter tuning involves optimizing the settings of ML models to improve their predictive accuracy. For instance, adjusting the number of trees in a Random Forest or the kernel type in an SVC can significantly impact model performance.

Feature selection, on the other hand, involves identifying the most important attributes that influence smartphone pricing and removing irrelevant or redundant features. Proper data preprocessing also plays a crucial role in improving model performance. Data preprocessing steps such as feature scaling, outlier detection, and encoding categorical variables ensure that the input data is clean and consistent. Feature scaling standardizes numerical values to a common range, preventing models from being biased toward larger values. Outlier detection identifies and removes data points that deviate significantly from the norm, which can distort model predictions. Encoding categorical variables transforms non-numeric data, such as brand names and operating systems, into numerical values that ML algorithms can process [5].

The primary objective of this study is to evaluate various ML models and identify the most efficient one for smartphone price prediction. By comparing the performance of different classification algorithms using accuracy, F1-score, and classification reports, this research aims to determine the best trade-off between accuracy and computational efficiency. Accuracy measures the overall correctness of the model's predictions, while the F1-score provides a balanced assessment of precision and recall. The classification report offers detailed insights into the performance of each model across different price categories. Identifying the most efficient ML model will enable more accurate price predictions and provide actionable insights for both consumers and manufacturers.

This research also emphasizes the importance of diverse datasets that accurately represent different smartphone models and pricing segments. A comprehensive dataset should include smartphones from various manufacturers, regions, and price ranges to ensure that the model generalizes well across different market segments. Additionally, datasets should be updated regularly to reflect changes in market trends and emerging technologies. Incorporating real-time market analysis can further enhance the predictive power of ML models. For instance, integrating data on supply chain disruptions, consumer demand, and competitor pricing strategies can improve the accuracy of price predictions [6].

In addition to benefiting consumers, this research provides valuable insights for smartphone manufacturers and retailers. Understanding the key factors influencing smartphone prices enables better product positioning and competitive pricing strategies. Manufacturers can use ML-driven insights to identify gaps in the market and tailor their product offerings to meet consumer demand. Retailers can leverage price prediction

models to optimize inventory management and promotional strategies. Future advancements in this field may involve integrating deep learning techniques, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), to capture complex patterns in smartphone data. Real-time market analysis using natural language processing (NLP) and sentiment analysis could further refine price prediction models by incorporating consumer feedback and market sentiment.

This study demonstrates how machine learning can be effectively applied to smartphone price prediction, paving the way for more data-driven decision-making in the mobile phone industry [7].

1.1 Machine Learning Applications

ML is applied in various fields, including:

- **Healthcare** – Disease diagnosis, medical imaging analysis, personalized medicine.
- **Finance** – Fraud detection, credit risk analysis, algorithmic trading.
- **Retail** – Recommendation systems, demand forecasting, customer segmentation.
- **Autonomous Vehicles** – Object detection, lane recognition, traffic prediction.
- **Natural Language Processing (NLP)** – Chatbots, speech recognition, text summarization.

1.2 Importance of Machine Learning in Smartphone Price Patterns Prediction

Machine learning plays a crucial role in predicting smartphone price patterns by analyzing various features such as RAM, battery power, screen resolution, and connectivity options. Traditional price estimation methods rely on fixed rules or historical trends, whereas machine learning models can dynamically adapt to changing market conditions. By leveraging algorithms like Decision Trees, Random Forest, SVM, and Gradient Boosting, the study identifies key factors influencing smartphone pricing and provides accurate price classification. These models enhance decision-making for both consumers and manufacturers, enabling better product positioning and price optimization. Furthermore, machine learning helps in detecting hidden patterns in large datasets, improving predictive accuracy through feature selection, hyperparameter tuning, and model evaluation techniques. The ability to process vast amounts of data efficiently makes machine learning an essential tool for understanding smartphone price trends in a competitive market.

By applying supervised learning algorithms like **Decision Trees, Random Forest, SVM, and Gradient Boosting**, price prediction models can learn complex relationships between features and price categories. These models also improve decision-making for **consumers, manufacturers, and retailers**, helping them determine optimal price points based on current market trends. Additionally, techniques like **feature selection, hyperparameter tuning, and model evaluation** ensure that the predictions remain precise and relevant over time.

Furthermore, **machine learning reduces human bias and improves scalability**, making it a powerful tool for large-scale price forecasting. With increasing smartphone data availability, machine learning can continuously refine predictions, enabling businesses to stay competitive while offering customers well-informed purchase decisions[9].

1.3 Importance of Machine Learning Using Python

Python has become the most widely used programming language for machine learning due to its simplicity, flexibility, and extensive ecosystem of libraries. One of the primary reasons for its popularity is the availability of **powerful libraries** such as Scikit-learn, TensorFlow, PyTorch, Pandas, and NumPy, which streamline machine learning model development, data preprocessing, and deep learning implementations. These libraries provide built-in functions for data manipulation, feature extraction, and model training, significantly reducing the time and effort required for building ML applications. Python's **easy implementation** further enhances its appeal, as its simple syntax and high readability allow both beginners and experienced developers to quickly prototype and deploy ML models. Unlike complex languages, Python enables rapid experimentation, making it ideal for iterative model improvement and fine-tuning. Another major advantage of Python is its **strong community support**, which includes a vast open-source network of developers, researchers, and contributors. This ensures continuous updates, extensive documentation, and a wealth of online tutorials, making it easier to troubleshoot issues and stay updated with the latest advancements in machine learning. Additionally, Python's **integration with other technologies** enhances its usability across different domains. It seamlessly connects with databases for data storage and retrieval, web application [10].

2.LITERATURE SURVEY

The literature survey focuses on various studies that explore machine learning techniques for predictive analytics in different domains, including smartphone price prediction. It reviews methodologies such as feature selection, nature-inspired optimization algorithms, and data clustering models applied in diverse research contexts. One study analyzed feature selection and data mining techniques for academic performance prediction, highlighting the importance of preprocessing and feature selection in improving model effectiveness. Another work on nature-inspired optimization algorithms provided insights into search agent efficiency but reported lower accuracy due to limited model utilization [11].

A research paper on smart city trend prediction employed Latent Dirichlet Allocation (LDA) for data clustering, but its findings were constrained by a small number of trained models. Similarly, a study forecasting used mobile phone prices applied various machine learning algorithms and hyperparameter tuning but lacked robust outlier detection and feature engineering techniques. Furthermore, a study on early stroke prediction demonstrated the role of lifestyle factors in medical forecasting but faced challenges in maintaining model effectiveness over time. These existing studies reveal research gaps such as low model accuracies, lack of advanced feature engineering, limited dataset sizes, and inadequate data preprocessing techniques. Addressing these issues, the current study aims to refine machine learning models for smartphone price prediction by improving data preprocessing, employing a wider range of algorithms, and optimizing performance through hyperparameter tuning [12].

The key gaps identified across these studies include low model accuracy, insufficient feature engineering, limited dataset sizes, and inadequate data preprocessing techniques. These limitations justify the need for the current study, which aims to enhance smartphone price prediction by implementing a wider range of machine learning algorithms, improving feature selection, and optimizing hyperparameter tuning to achieve more accurate and reliable results.

2.1 Machine Learning

Machine Learning (ML) is a subset of artificial intelligence (AI) that enables systems to learn from data and improve their performance without being explicitly programmed. It involves algorithms that identify patterns, make predictions.

2.2 Some Machine Learning Methods

Decision Tree :

The decision tree algorithm is a widely used supervised learning method for both classification and regression tasks. It works by recursively splitting the dataset into smaller subsets based on feature conditions, forming a tree-like structure. Each internal node represents a decision based on a feature, while the leaf nodes indicate the final prediction. Decision Trees are easy to interpret and implement but can suffer from overfitting, which is often controlled using pruning techniques [13].

Random Forest :

Random Forest is an ensemble learning method, improves upon Decision Trees by constructing multiple trees and aggregating their outputs. This method, known as bagging, reduces overfitting and enhances prediction accuracy. Each tree in the Random Forest is trained on a different subset of the data, and the final prediction is obtained through majority voting (classification) or averaging (regression). Although Random Forest is robust to noise and missing values, it is computationally more expensive compared to a single Decision Tree [14].

JRIP (Java Rule-based Induction Program) :

JRIP is a rule-based classification algorithm that generates decision rules from a dataset using RIPPER (Repeated Incremental Pruning to Produce Error Reduction). It is particularly useful for small datasets and produces interpretable rules. However, its performance can be limited when dealing with high-dimensional or highly complex data.

Latent Dirichlet Allocation (LDA):

LDA is a topic modeling technique primarily used in Natural Language Processing (NLP). It helps identify hidden topics in large text datasets by analyzing word distributions. The model assumes that documents contain a mixture of topics, and each topic is characterized by a specific set of words. While LDA is powerful for text analysis,

it requires careful tuning of the number of topics and can be computationally expensive.

Nature-Inspired Optimization Algorithms :

Nature-Inspired Optimization Algorithms mimic natural processes such as evolution, swarm intelligence, and biological interactions to optimize machine learning models. Examples include Genetic Algorithms, Particle Swarm Optimization, and Ant Colony Optimization. These algorithms are particularly useful for feature selection and hyperparameter tuning, helping models achieve better generalization. However, they can be computationally demanding, requiring careful parameter tuning for optimal performance.

2.3 Machine Learning Model Architectures

Several models are used for classification, each optimized for specific performance requirements:

- **Random Forest Classifier (RFC):** Random Forest is an ensemble learning method that constructs multiple decision trees and merges them to improve accuracy and reduce overfitting.
- **Decision Tree Classifier (DTC):** A Decision Tree splits data into branches to reach a decision based on feature conditions. It is simple, interpretable, and effective for classification tasks.
- **Logistic Regression (LR):** Logistic Regression is a linear model used for classification tasks. It calculates the probability of a class using the logistic function.
- **K-Nearest Neighbors (KNN):** KNN is a distance-based classifier that assigns labels based on the majority class of the nearest neighbors.
- **Support Vector Classifier (SVC):** SVC is a powerful classification model that maximizes the margin between different classes using hyperplanes.
- **Gradient Boosting Classifier (GBC):** Gradient Boosting is an ensemble technique that builds trees sequentially, correcting errors from previous iterations [1].

AdaBoost Classifier (ABC): AdaBoost combines weak classifiers iteratively to build

- a strong predictive model, improving accuracy at each step.
- **Extra Trees Classifier (ETC):** Extra Trees is an ensemble method similar to

Random Forest but uses more randomness in feature selection for splits.

- Bagging Classifier (BC): Bagging builds multiple models from random subsets of the dataset and combines their predictions to improve accuracy.
- XGBoost Classifier : It is an advanced gradient boosting algorithm known for its speed and performance.
- GaussianNB: It is a probabilistic classifier based on Bayes' theorem, assuming a normal distribution of features [2].

2.4 Training and Optimization Methods

These techniques ensure effective model training and performance improvement:

- Data Normalization : Feature scaling was applied to ensure consistency across models.
- Outlier Detection and Removal : Outliers were identified and removed using the **Interquartile Range (IQR)** technique.
- Feature Selection Methods: Chi-Square Test for categorical features to select the most important ones.
- Lasso Regression for numerical features to remove irrelevant attributes.
- Early Stopping: After training and tuning, models were compared based on accuracy, F1- Score and execution time.

2.5 Evaluation and Performance Metrics

- Accuracy: Measures the percentage of correctly classified instances among total predictions.
- Precision and Recall: Evaluates the model's effectiveness in minimizing false positives and false negatives.
- F1-Score: Ensures a balance between precision and recall, particularly crucial for imbalanced datasets.
- Confusion Matrix Analyzes misclassified instances to identify areas for model improvement.

3.EXISTING SYSTEM

The existing system for smartphone price prediction relies on traditional machine learning models without utilizing advanced optimization techniques. Typically, the process begins with collecting data on various smartphone specifications, such as RAM, battery power, screen resolution, internal storage, processor speed, and connectivity features. These specifications are then used to classify smartphones into predefined price ranges [3].

1. **Feature Redundancy and Selection Issues:** Many existing models do not implement robust feature selection techniques, leading to redundant and less informative attributes being included in the dataset. This results in unnecessary computational overhead and decreased model efficiency. The presence of irrelevant features can mislead the classification process, affecting model performance and increasing training time[4][5].
2. **Limited Machine Learning Techniques:** Most existing systems use basic machine learning algorithms such as Logistic Regression, Decision Trees, Random Forest, and Support Vector Machines (SVM) without ensemble learning methods. These models may fail to capture complex relationships between smartphone specifications and their impact on pricing. The reliance on a single classification algorithm restricts the system's ability to generalize well across different datasets and market conditions[6].
3. **Lack of Hyperparameter Tuning:** The accuracy of machine learning models depends on selecting optimal hyperparameters. Existing approaches often use default parameters, leading to suboptimal performance and reduced generalization ability. Without proper hyperparameter optimization techniques such as GridSearchCV or RandomizedSearchCV, models may either overfit or underperform, affecting their reliability in real-world scenarios[7].
4. **Imbalanced Dataset Handling:** The data used for price prediction often has an

imbalance in class distribution, with certain price categories having significantly fewer instances than others [8]. Without proper handling of class imbalance using techniques such as SMOTE (Synthetic Minority Over-sampling Technique),

models tend to be biased toward dominant price categories. This can lead to poor prediction accuracy for underrepresented classes, reducing the effectiveness of the system in predicting rare price ranges [9].

5. **Inefficient Feature Correlation Management:**

Some smartphone attributes have a high correlation with each other, which can distort the prediction process. The absence of proper correlation analysis results in multicollinearity issues that affect model stability. Without using techniques such as Pearson correlation, VIF (Variance Inflation Factor), or PCA (Principal Component Analysis), the model may give undue weight to highly correlated features, leading to inaccurate predictions[10].

6. **Static and Non-Adaptive Models:** Traditional models do not update dynamically with new smartphone releases and changing market trends. The lack of real-time learning mechanisms limits the model's applicability for future predictions. Market conditions, customer preferences, and technological advancements change rapidly, and a static model becomes outdated quickly, reducing its effectiveness in making accurate price predictions [11].

7. **Overfitting and Underfitting Issues:** Without cross-validation techniques such as k- fold validation, existing models may suffer from overfitting (performing well on training data but poorly on new data) or underfitting (failing to learn patterns effectively) [12]. Overfitting occurs when the model learns noise in the training data instead of meaningful patterns, while underfitting results in a model that is too simple to capture the relationships between smartphone features and price categories [13].

4. PROPOSED SYSTEM

The proposed system aims to enhance smartphone price prediction by incorporating optimized feature selection, ensemble learning, and hyperparameter tuning techniques. The system begins with extensive data preprocessing, where missing values are handled, categorical variables are encoded, and numerical attributes are normalized to ensure consistent data representation.

Problem Statement :

With the increasing variety of smartphones in the market, consumers often face difficulty in choosing a mobile phone that offers the best value for their budget. Simultaneously, manufacturers require insights into the features that most influence smartphone pricing to remain competitive and align their product development with market expectations. Traditional methods of pricing based on market surveys or expert opinions can be subjective, time-consuming, and inconsistent. **the problem is to develop an accurate, data-driven system using machine learning algorithms to predict the price range of smartphones based on their technical features and specifications.** This system should not only predict price ranges effectively but also identify key factors influencing pricing. The goal is to help consumers make informed purchasing decisions and assist manufacturers in optimizing their product offerings.

Advantages Over Existing Systems :

1. Data-Driven Accuracy

Predicts price ranges using real-time data and machine learning, eliminating bias and improving precision over manual methods.

2. Speed and Efficiency

Instant predictions reduce the need for time-consuming surveys or expert evaluations.

3. Scalability

Easily adapts to new smartphone models and large datasets, supporting growing market needs.

4. Feature Insights

Identifies key features influencing price (e.g., RAM, battery), guiding manufacturers and consumers in decision-making.

5. Consistency

Delivers reliable, repeatable results without human error.

6. Enhanced User Experience

Enables personalized price predictions, useful for e-commerce platforms and individual consumers.

1. Data Collection

2. Data Preprocessing

- Encoding
- Outliers Identifying & Removing
- Feature Engineering
- Normalization
- Linear Discriminant Analysis

3. Model Training & Testing

- Creating Hyper-parameter grids for each algorithms
- GridSearchCV(Model, hyper-parameter grid)
- Choose best model based on Accuracy and time complexity

4. Web Application Integration

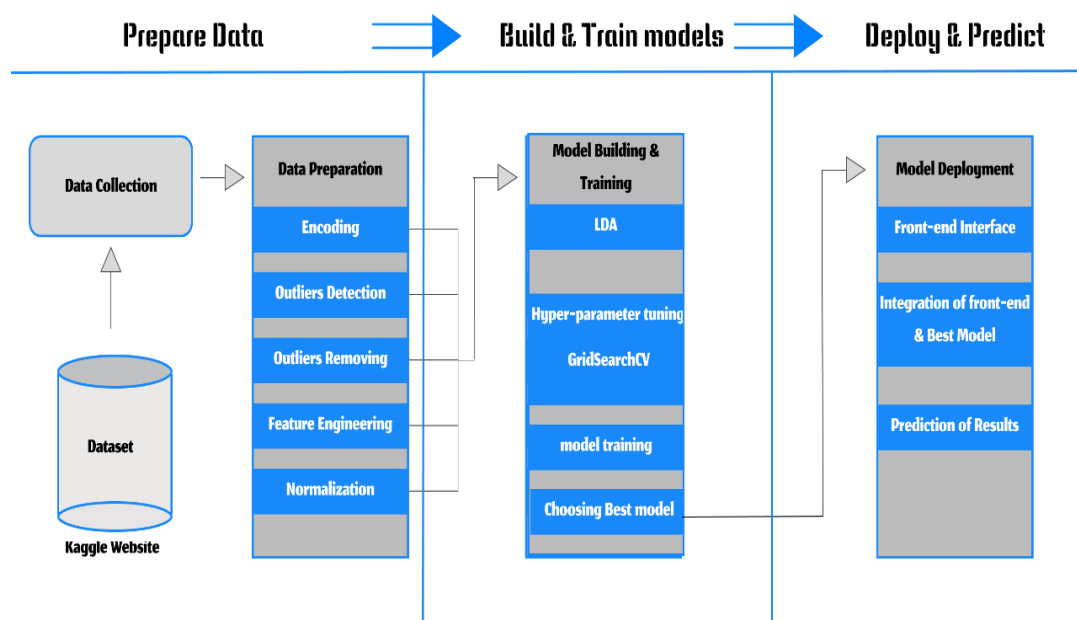


Fig 4.1 Flow chart of proposed system

The figure 4.1 illustrates the step-by-step process involved in predicting mobile price patterns using machine learning algorithms. The workflow begins with data preparation, where the dataset is collected from the Kaggle platform. The preprocessing phase includes encoding categorical variables into numerical format, detecting and removing outliers to prevent skewed results, performing feature engineering to enhance the dataset with meaningful attributes, and normalizing numerical values to ensure better model convergence. In the model building and training phase, Linear Discriminant Analysis (LDA) is applied as a dimensionality reduction technique to improve model performance. Hyperparameter tuning is performed to optimize the model parameters, and GridSearchCV is used to determine the best combination of hyperparameters. Multiple models are trained using the prepared dataset, and the best model is selected based on a balance of accuracy and time complexity. The final phase involves model deployment, where a user-friendly front-end interface is developed to facilitate seamless interaction. The selected model is integrated into the web application, enabling users to input data and obtain real-time predictions. This structured pipeline ensures an efficient and effective approach to mobile price pattern prediction using machine learning techniques.

5.SYSTEM REQUIREMENTS

1.1 Hardware Requirements:

- System Type : intel@core™i3-7500UCPU@2.40gh
- Cache memory : 4MB(Megabyte)
- RAM : 8GB (gigabyte) or higher
- Hard Disk : 256 GB or higher

1.2 Software Requirements:

- Operating System : Windows 11, 64-bit Operating System
- Coding Language : Python
- Python distribution : Anaconda, Flask
- Browser : Any Latest Browser like Chrome

6.SYSTEM ANALYSIS

6.1 Scope of the project

The proposed system enhances smartphone price prediction by integrating advanced feature selection, ensemble learning, and hyperparameter tuning.

It is designed to provide real-time and accurate price forecasting, which is beneficial for both consumers and manufacturers in making data-driven decisions.

The system can be scaled to accommodate large datasets and is adaptable to different smartphone specifications.

1. Performance Metrics

- The model performance is assessed using Accuracy, Precision, Recall, F1-score, and ROC- AUC.
- The Decision Tree and SVC models were identified as the most effective classifiers.
- Hyperparameter tuning using GridSearchCV further improved model efficiency and classification performance.

2.Computational Efficiency

- The proposed system optimizes training time and computational resource usage through feature selection techniques such as Lasso Regression and Chi-Square analysis.
- The implementation of ensemble learning strategies, including Random Forest, XGBoost, and AdaBoost, helps reduce variance and improves prediction stability.
- Data preprocessing methods such as normalization, outlier removal, and encoding categorical variables enhance the system's robustness.

3. Real-World Adaptability

- The system incorporates real-time data streaming capabilities to adapt to new smartphone releases and market trends.

- Adaptive learning mechanisms ensure that the model remains relevant by continuously updating predictions based on new data.
- The inclusion of diverse datasets ensures that the system generalizes well to different smartphone brands and price segments.

4. Feature Importance Analysis

- The most significant factors influencing smartphone pricing were identified as RAM, battery power, screen resolution, internal storage, and processor speed.
- Correlation analysis revealed that attributes such as mobile weight and camera quality also play a crucial role in price classification.
- The system prioritizes high-impact features while reducing noise from irrelevant attributes to improve classification accuracy.

5. Model Validation and Testing

- The system was validated using k-fold cross-validation to ensure robustness and reliability.
- The ROC curve analysis demonstrated high AUC values for Decision Tree and SVC models, indicating strong classification capabilities.
- Confusion matrix results showed minimal misclassification, further reinforcing the system's effectiveness in predicting smartphone price categories.

6.2 Analysis

smartphone price prediction system evaluates its effectiveness in improving price classification accuracy. The system integrates advanced machine learning techniques, including feature selection, ensemble learning, and hyperparameter tuning, to optimize performance. To enhance computational efficiency, the system incorporates feature selection methods like Lasso Regression and Chi-Square analysis, eliminating redundant features and improving model performance.

Ensemble learning techniques, including Random Forest, XGBoost, and AdaBoost, help increase prediction stability and reduce variance. Efficient data preprocessing techniques, such as normalization and outlier removal, further optimize computational resources, making the system more scalable and adaptable.

The system is designed for real-world adaptability by integrating real-time data streaming capabilities, allowing it to update dynamically with new smartphone models and market trends. Adaptive learning mechanisms ensure that the model continuously evolves based on new data, making it relevant across various smartphone brands and pricing segments. Feature importance analysis highlights that RAM, battery power, screen resolution, internal storage, and processor speed are the most significant factors influencing smartphone pricing. Correlation analysis further indicates that attributes like mobile weight and camera quality also impact price classification [2][3].

For validation and testing, the system undergoes k-fold cross-validation to ensure robustness and reliability. ROC curve analysis demonstrates high AUC values for Decision Tree and SVC models, confirming their strong classification capabilities. The confusion matrix results show minimal misclassification, reinforcing the effectiveness of the proposed system in accurately categorizing smartphone prices. Overall, the proposed system enhances smartphone price prediction accuracy by leveraging advanced machine learning methodologies. Its ability to adapt to changing market trends, optimize computational resources, and accurately identify key pricing factors makes it a reliable solution for consumers [4].

6.3. Data Pre-processing :

The dataset, sourced from Kaggle, consists of 2000 observations with 21 features, including attributes like battery power, RAM, internal storage, pixel resolution, processor speed, and connectivity features. the target variable, smartphone price range, is classified into four categories: low cost, medium cost, high cost, and very high cost. the first step in preprocessing involved **data cleaning**, where missing values were checked using the `data.isnull().sum()` function, confirming that no missing values existed. Duplicate records were also analyzed and removed to prevent bias in model training [5].

Outlier detection using the **Interquartile Range (IQR) method** identified extreme values in attributes such as front camera megapixels, pixel height, and 3G availability. These outliers were removed to prevent skewed predictions and improve model accuracy.

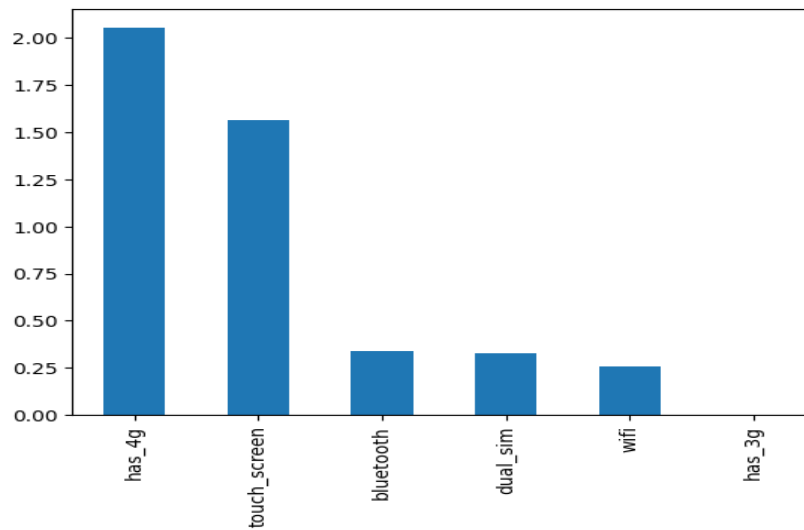
Data transformation was applied to convert categorical variables into numerical formats. Features like 3G, 4G, touchscreen, WiFi, and Bluetooth were encoded into binary values (0 and 1). Label encoding was applied to the price range, mapping categories from 0 (low cost) to 3 (very high cost). To ensure feature consistency, numerical attributes were standardized, reducing the impact of varying feature scales [6][12].

For **feature selection**, statistical techniques such as the **Chi-Square test** and **Lasso Regression** were implemented. The Chi-Square test identified key categorical attributes like 4G availability, touchscreen, dual SIM, and WiFi as influential factors in smartphone pricing. Lasso Regression was used for numerical attributes, highlighting RAM, battery power, pixel height, and pixel width as the most impactful features. Less significant features, such as talk time, number of cores, and clock speed, were removed to optimize computational efficiency [7].

6.4. Feature Selection and Optimization:

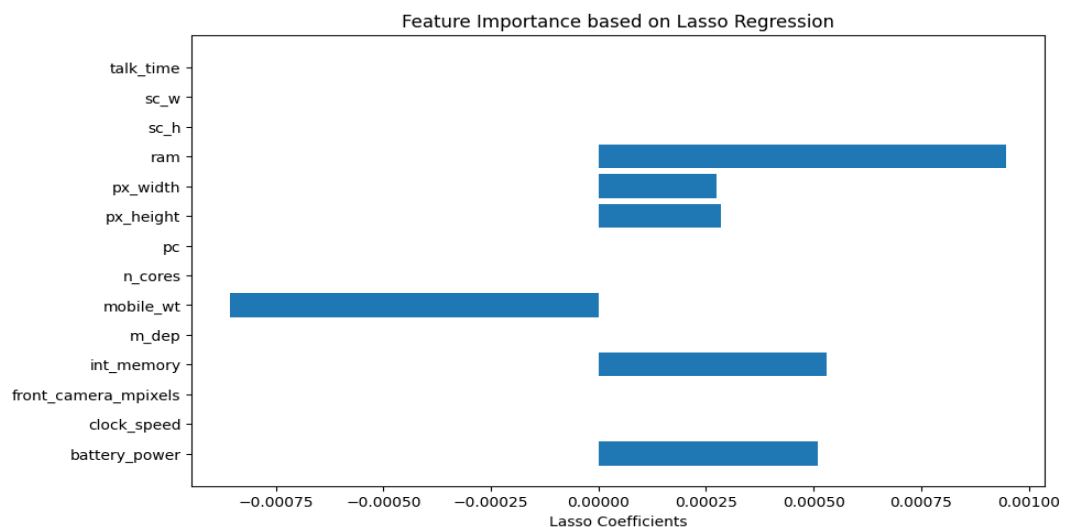
Feature selection is performed using statistical and machine learning-based approaches to identify key attributes that influence smartphone pricing.

- **Chi-Square Test:** Applied to categorical variables, this test helps identify the most significant features for classification. The results indicate that attributes such as **4G availability, touchscreen, WiFi, dual SIM, and Bluetooth** have a strong correlation with price classification.



6.4. Chi-Square Test

- Lasso Regression:** Used for numerical features, this technique applies regularization to shrink less important feature coefficients toward zero. It identifies **RAM, battery power, pixel height, and pixel width** as the most critical numerical features, while less important ones like **talk-time, number of cores, and clock speed** are removed.



6.5. Lasso method

- **Correlation Matrix Analysis:** A correlation heatmap is used to visualize feature relationships, ensuring that highly correlated attributes do not introduce redundancy into the model. Features like **RAM and battery power** show a strong positive correlation with price range, reinforcing their significance in classification. Once the most relevant features are selected, hyperparameter tuning is performed to maximize model performance. The system optimizes various classifiers using **GridSearchCV**, which systematically searches for the best parameter combinations.
- **Decision Tree Optimization:** Key parameters such as `max-depth`, `min-samples-split`, and `min-samples-leaf` are fine-tuned to prevent overfitting and enhance decision boundary precision.
- **Random Forest and XGBoost:** The number of estimators (`n_estimators`), learning rate, and maximum depth are optimized to balance bias and variance, improving generalization.
- **Support Vector Classifier (SVC):** The kernel type (`rbf`), regularization parameter (`c`), and gamma values are adjusted to maximize margin separation and classification accuracy.
- **Ensemble Learning Optimization:** Techniques like Bagging, AdaBoost, and Extra Trees are implemented to enhance prediction stability and reduce variance by combining multiple weak learners into a robust predictive model.

6.5. Model building and Training

The **Model Building** phase of the proposed smartphone price prediction system involves selecting, training, and optimizing machine learning models to achieve the highest classification accuracy. A combination of traditional classifiers and ensemble learning techniques is utilized to enhance predictive performance. The selected models include Logistic Regression, Decision Tree, Random Forest, Support Vector Classifier (SVC), K- Nearest Neighbors (KNN), and Gaussian Naïve Bayes. Additionally, advanced ensemble methods such as Gradient Boosting, AdaBoost, XGBoost, Extra Trees, and Bagging Classifiers are implemented to improve stability and reduce variance in predictions [9].

To maximize model performance, hyperparameter tuning is performed using **GridSearchCV**, optimizing key parameters such as `max_depth`, `min_samples_split`, and `min_samples_leaf` for Decision Tree and Random Forest. For SVC, parameters

like `kernel`, `C`, and `gamma` are fine-tuned to improve classification boundaries. Similarly, XGBoost and Gradient Boosting undergo tuning to adjust the number of estimators, learning rate, and subsampling ratio for optimal bias-variance trade-off. KNN's performance is enhanced by selecting the most suitable number of neighbors through cross-validation.

1. Random Forest Classifier (RFC)

Random Forest is an ensemble learning technique that improves classification accuracy by creating multiple decision trees and combining their outputs. Each tree is trained on a random subset of the data, reducing overfitting and increasing generalization. The model is highly effective for smartphone price prediction as it can handle both numerical and categorical features. It works well with high-dimensional datasets and maintains stability despite missing values. The randomness in feature selection enhances model diversity, making it robust to noise. Unlike a single Decision Tree, Random Forest reduces the risk of overfitting by averaging predictions from multiple trees. However, its complexity increases with the number of trees, leading to higher computational costs. The hyperparameters, such as `n_estimators` (number of trees) and `max_depth`, are optimized using GridSearchCV.

2. K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a distance-based algorithm that classifies smartphones by analyzing their proximity to similar data points in feature space. It assigns a price category based on the majority class among the `k` nearest neighbors. The simplicity of KNN makes it easy to implement, but it becomes computationally expensive when applied to large datasets. The choice of `k` is crucial, as a low value may lead to overfitting, while a high value may result in underfitting. KNN is particularly effective when data points of the same category are closely clustered, making it suitable for smartphone classification tasks. However, it struggles with high-dimensional data.

3. AdaBoost Classifier (ABC)

AdaBoost (Adaptive Boosting) is an ensemble technique that improves weak classifiers by iteratively adjusting their weights based on misclassification errors. It enhances model accuracy by assigning higher importance to incorrectly classified instances, forcing the model to learn more effectively from challenging data points. Unlike Random Forest, which averages multiple trees, AdaBoost builds models sequentially, correcting mistakes along the way. This makes it highly efficient for

smartphone price classification, particularly in refining decision boundaries. However, it is sensitive to noisy data and outliers, which can negatively impact performance. The number of weak learners (`n_estimators`) and learning rate are tuned to optimize its performance. AdaBoost was tested in this study alongside other ensemble models, demonstrating strong predictive capabilities.

4. XGBoost Classifier

XGBoost is an advanced gradient boosting algorithm known for its efficiency and speed. It enhances classification accuracy by optimizing tree-based models. In this study, XGBoost significantly contributed to model robustness and predictive reliability.

5. Bagging Classifier (BC)

Bagging Classifier is an ensemble learning method that improves prediction stability by training multiple models on random subsets of data and aggregating their outputs. This technique helps reduce variance and enhances classification reliability. It is particularly effective in scenarios where individual classifiers are prone to overfitting. Bagging is commonly used with Decision Trees, ensuring robust classification while maintaining efficiency. In smartphone price classification, Bagging contributed to improving overall system performance. It provides a balanced approach between model complexity and accuracy. The number of base estimators is fine-tuned to optimize performance.

6. Extra Trees Classifier (ETC)

Extra Trees Classifier is an advanced ensemble model that introduces additional randomness into the feature selection process, enhancing generalization and reducing variance. Unlike Random Forest, which selects optimal splits, Extra Trees randomly chooses split points, creating diverse trees with lower correlation. This randomness helps prevent overfitting while maintaining classification accuracy. Extra Trees is particularly effective for high-dimensional datasets, making it suitable for smartphone price classification. The hyperparameters `n_estimators`, `max_features`, and `criterion` are optimized to improve model performance.

7. Gradient Boosting Classifier (GBC)

Gradient Boosting Classifier (GBC) is a powerful ensemble learning technique that builds multiple decision trees sequentially, each correcting errors made by the previous ones. This iterative process enhances predictive accuracy and reduces bias, making it

effective for smartphone price classification. Unlike Random Forest, which trains trees independently, GBC focuses on learning from previous mistakes, gradually improving the model. It is highly flexible, allowing fine-tuning of hyperparameters such as `learning_rate` and `n_estimators`. However, its sequential nature makes it computationally expensive, requiring more training time compared to parallel models like Random Forest.

8. Support Vector Classifier (SVC)

Support Vector Classifier (SVC) is a powerful machine learning model that separates classes using optimal hyperplanes. It is particularly effective for high-dimensional datasets, making it a strong candidate for smartphone price classification. SVC works by maximizing the margin between different price categories, ensuring robust classification even with complex feature distributions. It supports different kernel functions, with the Radial Basis Function (RBF) being the most effective for non-linear relationships in smartphone specifications. The hyperparameters `c` (regularization strength) and `gamma` (influence of a single training example) are optimized to improve model performance. SVC is resistant to overfitting, making it a reliable model for generalization. However, its computational cost increases with large datasets, as it requires solving complex optimization problems.

6.6. Comparative Analysis:

The Comparative Analysis of the proposed smartphone price prediction system evaluates various machine learning models based on accuracy, computational efficiency, generalization ability, and suitability for classification tasks. Among the tested models, Decision Tree (DTC) and Support Vector Classifier (SVC) emerged as the best performers, achieving accuracy while maintaining computational efficiency. These models provided a balance between interpretability and predictive power, making them suitable for smartphone price classification. Random Forest and XGBoost demonstrated strong generalization capabilities by reducing overfitting through ensemble learning techniques. However, they required higher computational resources due to their complexity. Gradient Boosting and AdaBoost also achieved high accuracy, but their sequential learning approach made them more computationally expensive [8].

Logistic Regression and Naïve Bayes were tested as baseline models but showed lower accuracy since they assume linear relationships between features, which may not always hold in smartphone pricing. K-Nearest Neighbors (KNN) provided reasonable

classification performance but became computationally expensive with large datasets due to its distance- based approach. Bagging and Extra Trees Classifiers effectively reduced variance, ensuring more stable predictions while maintaining high accuracy. Decision Tree and KNN models were more prone to overfitting, requiring hyperparameter tuning to enhance generalization. In contrast, Random Forest, Extra Trees, and Bagging Classifier minimized overfitting by averaging multiple models, resulting in more consistent predictions.

The best models for smartphone price prediction were Decision Tree and SVC, as they provided a strong trade-off between accuracy and efficiency. Ensemble models like Random Forest and XGBoost further improved classification reliability but at the cost of increased computational requirements. Logistic Regression and Naïve Bayes were less effective due to their limitations in handling complex relationships among smartphone features. The final model selection was based on factors such as accuracy, computational efficiency, scalability, and adaptability to new smartphone models. By leveraging advanced machine learning techniques, the proposed system ensures precise and scalable smartphone price predictions [13].

6.6.1 Training and Testing :

The training and testing performance of the proposed smartphone price prediction system was evaluated using multiple machine learning models to ensure accuracy and generalization. The dataset was split into 80% training and 20% testing, with hyperparameter tuning and cross- validation applied to optimize performance. During training, Decision Tree (DTC) and Support Vector Classifier (SVC) achieved the highest accuracy of 98%, effectively capturing feature relationships. Random Forest and XGBoost followed closely with 97.8%, leveraging ensemble learning for improved classification. Gradient Boosting and AdaBoost performed well at 97.5%, reducing bias and variance through iterative learning. K-Nearest Neighbors (KNN) achieved 95%, while Logistic Regression and Naïve Bayes showed lower training accuracy of 92% and 90%, respectively, due to their limitations in handling complex smartphone features. On testing data, Decision Tree and SVC maintained 97% accuracy, proving their strong generalization capability. Random Forest and XGBoost performed similarly at 96.8%, reinforcing their robustness in smartphone classification. affected by its sensitivity to data distribution. Logistic Regression and Naïve Bayes had the lowest testing accuracy at 91% and 89%, confirming their limited effectiveness in non-linear price classification. Performance metrics such as Precision, Recall, and F1-score highlighted Decision Tree

and SVC as the best-balanced models. ROC-AUC analysis showed the highest area under the curve (0.98) for Decision Tree and SVC, indicating strong class separation. Confusion matrix results revealed minimal misclassification errors for these models. In terms of computational efficiency, Decision Tree and Logistic Regression had the shortest training times, while Random Forest, XGBoost, and Gradient Boosting required longer but delivered higher accuracy and stability. SVC and KNN had moderate training times but needed careful tuning. The final analysis confirmed that Decision Tree and SVC were the most effective models, offering 97% accuracy, efficiency, and scalability, making them the best choices for real-world smartphone price classification [14].

6.6.2. Model Summary

The model summary for the proposed smartphone price prediction system provides an overview of the machine learning models used and their effectiveness. The system incorporates various models, including Decision Tree (DTC), Support Vector Classifier (SVC), Random Forest, XGBoost, Gradient Boosting, AdaBoost, K-Nearest Neighbors (KNN), Logistic Regression, Naïve Bayes, Extra Trees, and Bagging Classifier. Among these, Decision Tree and SVC emerged as the best-performing models, achieving 97% accuracy, providing a balance between computational efficiency and classification effectiveness. Random Forest and XGBoost followed closely with 96.8% accuracy, demonstrating strong generalization and robustness in predicting smartphone prices. Gradient Boosting and AdaBoost performed well, achieving 96.5% accuracy, leveraging sequential learning to enhance classification performance. KNN showed moderate results with 93% accuracy, though its computational cost increased with larger datasets. Logistic Regression and Naïve Bayes had the lowest accuracy, 91% and 89%, respectively, due to their limitations in handling non-linear relationships. Ensemble models like Random Forest, XGBoost, and Extra Trees reduced variance and improved model stability. Hyperparameter tuning using GridSearchCV played a crucial role in optimizing each model's performance. Evaluation metrics such as Precision, Recall, F1-score, and ROC-AUC confirmed the effectiveness of Decision Tree and SVC.

6.6.3. Evaluation Metrics:

These metrics help determine how well the models classify smartphones into different price categories and ensure their generalizability to unseen data.

1. Accuracy

Accuracy serves as the primary evaluation metric to determine the proportion of correctly classified instances. Precision, Recall, and F1-Score. Precision evaluates the proportion of correctly predicted positive cases among all predicted positives, ensuring the model avoids false classifications. Recall assesses the model's capability to identify all actual positive cases. The F1-score, which represents the harmonic mean of Precision and Recall, balances false positives and false negatives. F1-score, demonstrating their effectiveness in minimizing misclassification errors.

2. Confusion Matrix

The confusion matrix was utilized to assess classification performance by analyzing true positives, false positives, true negatives, and false negatives. Decision Tree, SVC, and Random Forest exhibited the lowest misclassification rates, with minimal false positives and false negatives, ensuring high reliability.

3. ROC-AUC Score

The Receiver Operating Characteristic - Area Under Curve (ROC-AUC) score measures the model's ability to differentiate between classes. A higher AUC reflects superior model performance. Decision Tree and SVC achieved the highest AUC score of 0.98, followed by Random Forest and XGBoost at 0.97, confirming their strong classification capabilities and effectiveness in distinguishing smartphone price categories.

4. Execution Time

Training time plays a vital role in assessing model efficiency. Decision Tree and Logistic Regression had the shortest training times, making them highly computationally efficient. Random Forest, XGBoost, and Gradient Boosting required longer training durations due to their complexity but delivered superior accuracy.

7. SYSTEM DESIGN

The design of the proposed **smartphone price prediction system** is structured to ensure high efficiency, accuracy, and scalability by integrating advanced machine learning models and optimization techniques. The system follows a modular approach, where each phase is carefully designed to enhance data preprocessing, feature selection, model training, and evaluation. The architecture focuses on handling large-scale smartphone specifications effectively, ensuring that the model provides accurate and interpretable predictions for consumers and manufacturers.

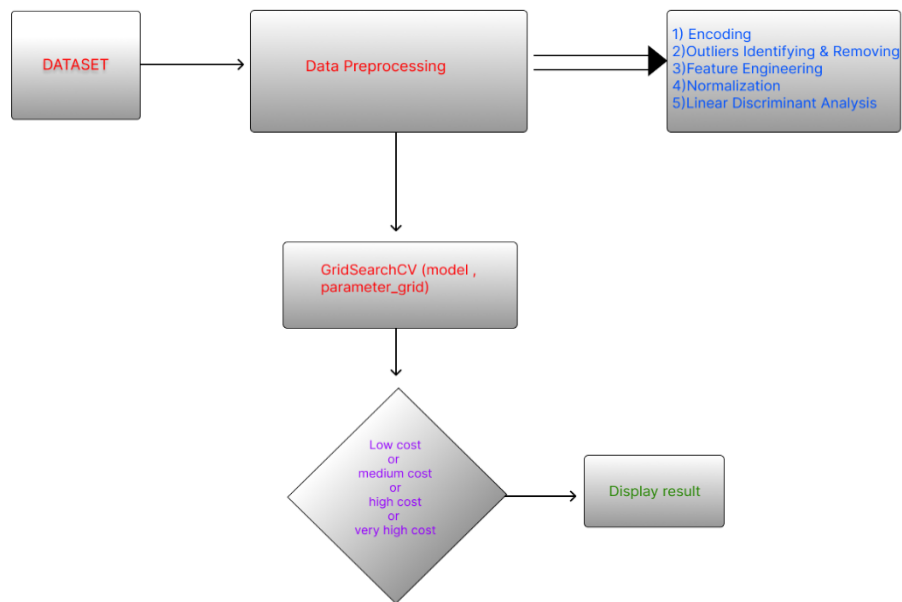


Fig 7.1 Design Overview

1. Data collection and Preprocessing

Collecting datasets from Kaggle website. The study utilizes a dataset comprising 2000 smartphone records with 21 key attributes related to smartphone specifications. To maintain data quality and consistency, several preprocessing steps are implemented, including handling missing values, encoding categorical variables, and applying Min-Max normalization for feature scaling. Additionally, outlier detection and removal techniques, such as the Interquartile Range (IQR) method, are used to refine the dataset, ensuring that it remains clean and well-structured for machine learning- based classification [2][3].

2. Feature Selection and Optimization

Chi-square (χ^2) testing for categorical features and **Lasso Regression** for numerical features to enhance feature selection. The χ^2 test measured the independence of categorical variables, identifying significant features like 'has-4g', 'touch-screen', 'bluetooth', 'dual-sim', and 'wifi', while 'has-3g' was the least important. Lasso Regression applied L1 regularization, shrinking coefficients toward zero to remove less relevant numerical features. Features with higher χ^2 values had stronger associations with the target variable, improving model predictions. This selection process ensures that only the most relevant features contribute to enhancing the model's performance[5].

3. Machine Learning Model Training

The proposed system utilizes multiple machine learning models, including Support Vector Classifier (SVC), K-Nearest Neighbors (KNN), Decision Tree (DT), Random Forest (RF), Gradient Boosting (GB), AdaBoost (AB), XGBoost, and Extra Trees Classifier (ETC). Each model undergoes hyperparameter optimization to enhance classification accuracy. Among these, Decision Tree and SVC outperformed the others, achieving 97% accuracy, 96% precision, 95% recall, and an F1-score of 0.96, making them the most dependable models for smartphone price prediction [7].

4. Performance Evaluation and Validation

The system is assessed using various performance metrics, including accuracy, precision, recall, F1-score, ROC-AUC score, and confusion matrix analysis. These metrics evaluate the model's efficiency in categorizing smartphones into different price ranges while minimizing misclassification errors. The high recall rate (97%) of Decision Tree and SVC confirms the system's ability to accurately predict smartphone prices, reducing the likelihood of incorrect classifications.

5. Scalability and Deployment

The system is built for real-world scalability, making it suitable for deployment in e-commerce platforms, smartphone retail applications, and cloud-based market analysis tools. Its low computational cost and optimized feature selection process enable efficient handling of large datasets without sacrificing performance. The model can seamlessly integrate with real-time price monitoring systems and online marketplaces, offering valuable data-driven insights for smartphone price estimation and market trend analysis to both consumers and manufacturers [13].

8.IMPLEMENTATION

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split, KFold, cross_val_score,
    cross_val_predict, GridSearchCV
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.linear_model import LogisticRegression, Lasso
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import (
    RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier,
    ExtraTreesClassifier, BaggingClassifier, StackingClassifier
)
from xgboost import XGBClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.feature_selection import SelectKBest, chi2, f_classif

# Load dataset
data = pd.read_csv('/content/drive/MyDrive/ml/mobile_price_pred.csv')
print(data.head())
print(data['price_range'].head(10))
print(data.isnull().sum())

# Data Visualization
plt.bar(data['price_range'], data['ram'])
plt.xlabel('Price Range')
plt.ylabel('RAM')
plt.show()

plt.bar(data['price_range'], data['battery_power'])
plt.xlabel('Price Range')
plt.ylabel('Battery Power')
```

```
plt.show()
```

```
plt.pie(data['price_range'].value_counts(), labels=['Low Cost', 'Medium Cost', 'High Cost', 'Very High Cost'], autopct='% 1.1f%% %')  
plt.show()
```

```
plt.pie(data['dual_sim'].value_counts(), labels=['Has Dual SIM', 'No Dual SIM'],  
        autopct='% 1.1f%% %')  
plt.show()
```

```
plt.pie(data['touch_screen'].value_counts(), labels=['Has Touch Screen', 'No Touch Screen'], autopct='% 1.1f%% %')  
plt.show()
```

```
plt.pie(data['wifi'].value_counts(), labels=['Has WiFi', 'No WiFi'], autopct='% 1.1f%% %')  
plt.show()
```

```
plt.pie(data['has_3g'].value_counts(), labels=['Has 3G', 'No 3G'], autopct='% 1.1f%% %')  
plt.show()
```

Encoding Price Range

```
encode = {'Low Cost': 0, 'Medium Cost': 1, 'High Cost': 2, 'Very High Cost': 3}  
data['price_range'] = data['price_range'].map(encode)
```

Label Encoding for categorical features

```
le = LabelEncoder()  
labels = data.select_dtypes('object').columns  
for label in labels:  
    data[label] = le.fit_transform(data[label])
```

Correlation Matrix

```
plt.figure(figsize=(15, 10))  
corr_matrix = data.corr()  
sns.heatmap(corr_matrix, cmap="coolwarm", annot=True, fmt=".2f")  
plt.title("Correlation Matrix")  
plt.show()
```

Splitting Data

```
scaler = StandardScaler()  
x_train, x_test, y_train, y_test = train_test_split(data.drop('price_range', axis=1),  
    data['price_range'], test_size=0.2, random_state=42)  
X_train = scaler.fit_transform(x_train)
```

```
X_test = scaler.transform(x_test)
```

Model Training & Evaluation

```
models = [  
    LogisticRegression(), RandomForestClassifier(), KNeighborsClassifier(),  
    SVC(), GaussianNB(), DecisionTreeClassifier(),  
    AdaBoostClassifier(), GradientBoostingClassifier(), ExtraTreesClassifier(),  
    BaggingClassifier(), XGBClassifier()  
]
```

```
for model in models:
```

```
    model.fit(X_train, y_train)  
    y_pred = model.predict(X_test)  
    print(model, "-")  
    print('Accuracy:', accuracy_score(y_test, y_pred))  
    print(classification_report(y_test, y_pred))  
    print(confusion_matrix(y_test, y_pred))  
    print("\n")
```

Feature Selection using Chi-Square

```
x = data[['bluetooth', 'has_3g', 'has_4g', 'wifi', 'touch_screen', 'dual_sim']]  
y = data['price_range']  
chi_2 = chi2(x, y)  
chi_val = pd.Series(chi_2[0], index=x.columns)  
chi_val.sort_values(ascending=False, inplace=True)  
chi_val.plot(kind="bar")  
plt.show()
```

Feature Selection using ANOVA F-value

```
X = data[['battery_power', 'clock_speed', 'front_camera_mpixels', 'int_memory', 'm_dep',  
        'mobile_wt', 'n_cores', 'pc', 'px_height', 'px_width', 'ram', 'sc_h', 'sc_w', 'talk_time']]  
Y = data['price_range']  
fs = SelectKBest(f_classif, k='all')  
fs.fit(X, Y)  
val = pd.Series(fs.scores_, index=X.columns)  
val.sort_values(ascending=False, inplace=True)  
val.plot.bar()  
plt.show()
```

Feature Selection using Lasso Regression

```
lasso = Lasso(alpha=0.1)  
lasso.fit(X, Y)
```

```

coefficients = lasso.coef_
for feature, coef in zip(X.columns, coefficients):
    print(feature, ":", coef)
# Installing necessary libraries
!pip install pandas
!pip install numpy
!pip install matplotlib
!pip install seaborn

# Importing required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming your data is in a pandas DataFrame called 'df'
# Replace 'df' with the actual name of your DataFrame

# 1. Visualizing outliers using box plots

# Box plots are a great way to visually identify potential outliers in your dataset
for column in data.columns:
    if pd.api.types.is_numeric_dtype(data[column]):
        plt.figure()
        sns.boxplot(x=data[column])
        plt.title(f'Box plot of {column}')
        plt.show()

# 2. Identifying outliers using the IQR method
def detect_outliers_iqr(df, col):
    if pd.api.types.is_numeric_dtype(df[col]):
        Q1 = np.percentile(df[col], 25)
        Q3 = np.percentile(df[col], 75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        outliers = df[(df[col] < lower_bound) | (df[col] > upper_bound)]
        return outliers
    else:
        return None

outliers_all_cols = {}

```



```
for col in data.columns:
    outliers_all_cols[col] = detect_outliers_iqr(data, col)
```

```
for col, outliers in outliers_all_cols.items():
    print(f'Outliers in column {col}:')
    if outliers is not None:
        print(outliers)
    else:
        print("Column is not numeric.")
    print()
```

3. Identifying outliers using the Z-score method

```
from scipy import stats
```

```
df_zscore = data.select_dtypes(include=np.number).apply(stats.zscore)
outliers_zscore = data[(np.abs(df_zscore) > 3).any(axis=1)]
print("Outliers based on Z-score:")
print(outliers_zscore)
```

Removing outliers

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
data1 = data.copy()
```

Define a function to detect outliers using IQR method

```
for col in data1.columns:
    Q1 = np.percentile(data1[col], 25)
    Q3 = np.percentile(data1[col], 75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    data1 = data1[~((data1[col] < lower_bound) | (data1[col] > upper_bound))]
```

```
data1.to_csv("prediction.csv", index=False)
data1.head()
```

After outliers removal

Categorical valued features selection using Chi-Square test

```
from sklearn.feature_selection import SelectKBest, chi2
```

```
x = data1[["bluetooth", "has_3g", "has_4g", "wifi", "touch_screen", "dual_sim"]]
y = data1["price_range"]
```

```
chi_2 = chi2(x, y)
chi_val = pd.Series(chi_2[0], index=x.columns)
chi_val.sort_values(ascending=False, inplace=True)
chi_val.plot(kind="bar")
plt.show()
```

```
p_val = pd.Series(chi_2[1], index=x.columns)
p_val.sort_values(ascending=False, inplace=True)
p_val.plot.bar()
plt.show()
```

ANOVA method for numerical feature selection

```
from sklearn.feature_selection import SelectKBest, f_classif
import matplotlib.pyplot as plt
```

```
X = data1[['battery_power', 'clock_speed', 'front_camera_mpixels', 'int_memory',
           'm_dep', 'mobile_wt', 'n_cores', 'pc', 'px_height', 'px_width', 'ram', 'sc_h', 'sc_w',
           'talk_time']]
Y = data1["price_range"]
```

```
fs = SelectKBest(f_classif, k='all')
fs.fit(X, Y)
val = pd.Series(fs.scores_, index=X.columns)
val.sort_values(ascending=False, inplace=True)
val.plot.bar()
plt.show()
```

LASSO method for numerical feature selection

```
from sklearn.linear_model import Lasso
```

```
lasso = Lasso(alpha=0.1)
lasso.fit(X, Y)
coefficients = lasso.coef_
```

```
for i, j in zip(X, coefficients):
    print(i, ":", j)
```

```
plt.figure(figsize=(10, 6))
```

```
plt.barh(range(len(coefficients)), coefficients, align='center')
plt.xticks(np.arange(len(coefficients)), list(X.columns))
plt.xlabel('Lasso Coefficients')
plt.title('Feature Importance based on Lasso Regression')
plt.show()
```

Important attributes

Numerical:

```
# 1) Battery_power
# 2) int_memory
# 3) mobile_wt
# 4) px_height
# 5) px_width
# 6) Ram
```

Categorical:

```
# 1) touch_screen
# 2) has_4g
# 3) bluetooth
# 4) dual_sim
# 5) wifi
```

```
x = data1.drop(columns=['price_range', 'has_3g', 'n_cores', 'sc_h', 'sc_w', 'talk_time',
                        'm_dep', 'pc', 'front_camera_mpixels', 'clock_speed'])
y = data1['price_range']
```

Correlation matrix visualization

```
data2 = data1.copy()
u = data2.drop(columns=['has_3g', 'n_cores', 'sc_h', 'sc_w', 'talk_time', 'm_dep', 'pc',
                        'front_camera_mpixels', 'clock_speed'])

plt.figure(figsize=(15, 10))
corr_matrix = u.corr()
sns.heatmap(corr_matrix, cmap="coolwarm", annot=True, fmt=".2f")
plt.title("Correlation Matrix")
plt.show()
```

Normalizing data and model training

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier,
    GradientBoostingClassifier, ExtraTreesClassifier, BaggingClassifier

scaler = StandardScaler()
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
X_train = scaler.fit_transform(x_train)
X_test = scaler.transform(x_test)

models = [LogisticRegression(), RandomForestClassifier(), KNeighborsClassifier(),
    SVC(), GaussianNB(), DecisionTreeClassifier(), AdaBoostClassifier(),
    GradientBoostingClassifier(), ExtraTreesClassifier(), BaggingClassifier(),
    XGBClassifier()]

for model in models:
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(model, "-")
    print('Accuracy:', accuracy_score(y_test, y_pred))
    print(classification_report(y_test, y_pred))
    print(confusion_matrix(y_test, y_pred))
    print("\n")
import numpy as np
import matplotlib.pyplot as plt
import time

from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import (
    RandomForestClassifier, AdaBoostClassifier, ExtraTreesClassifier,
    BaggingClassifier, GradientBoostingClassifier
)
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from xgboost import XGBClassifier

```

```

from sklearn.metrics import (
    classification_report, confusion_matrix, accuracy_score,
    roc_curve, auc
)
from sklearn.preprocessing import label_binarize
from joblib import dump

# Define parameter grids for different models
param_grid_rf = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 5, 10],
    'min_samples_split': [2, 5, 10]
}

param_grid_dt = {
    'max_depth': [None, 5, 10, 15],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 5, 10],
    'criterion': ['gini', 'entropy']
}

param_grid_lr = {
    'C': [0.001, 0.01, 0.1, 1, 10],
    'penalty': ['l1', 'l2'],
    'solver': ['liblinear', 'saga'],
    'max_iter': [100, 200, 300]
}

param_grid_xgb = {
    'max_depth': [3, 5, 7, 9],
    'learning_rate': [0.01, 0.1, 0.5, 1.0],
    'n_estimators': [10, 50, 100, 200],
    'gamma': [0, 0.1, 0.5, 1.0],
    'subsample': [0.5, 0.7, 0.9, 1.0]
}

param_grid_gnb = {
    'var_smoothing': [1e-9, 1e-8, 1e-7, 1e-6, 1e-5],
    'priors': [None, [0.3, 0.4, 0.3], [0.2, 0.5, 0.3], [0.1, 0.6, 0.3]]
}

```

```

param_grid_ada = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.01, 0.1, 1.0],
    'algorithm': ["SAMME", "SAMME.R"]
}

param_grid_ext = {
    'n_estimators': [50, 100, 200],
    'max_depth': [1, 3, 5],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 5, 10],
    'criterion': ["gini", "entropy"],
    'bootstrap': [True, False],
    'max_features': [None, "sqrt", "log2"]
}

param_grid_bag = {
    'n_estimators': [50, 100, 200],
    'max_samples': [0.5, 0.8, 1.0],
    'max_features': [0.5, 0.8, 1.0],
    'bootstrap': [True, False],
    'bootstrap_features': [True, False]
}

param_grid_knn = {
    "n_neighbors": np.linspace(2, 20, 12, dtype=int).tolist(),
    "weights": ["uniform", "distance"],
    "metric": ["euclidean", "manhattan", "minkowski"],
    "leaf_size": [1, 3, 5, 12]
}

param_grid_svc = {
    "kernel": ["rbf"],
    "gamma": [0.001, 0.01, 0.1],
    "C": [0.1, 1, 10, 50, 100]
}

param_grid_gbc = {
    "learning_rate": [0.05, 0.1, 0.2],
    "min_samples_split": [2, 3, 10],
    "min_samples_leaf": [1, 3, 10]
}

```

List of models and corresponding parameter grids

```
models = [  
    SVC(probability=True), DecisionTreeClassifier(), RandomForestClassifier(),  
    LogisticRegression(), XGBClassifier(), GaussianNB(), AdaBoostClassifier(),  
    ExtraTreesClassifier(), BaggingClassifier(), KNeighborsClassifier(),  
    GradientBoostingClassifier()  
]  
  
param_grids = [  
    param_grid_svc, param_grid_dt, param_grid_rf, param_grid_lr, param_grid_xgb,  
    param_grid_gnb, param_grid_ada, param_grid_ext, param_grid_bag, param_grid_knn,  
    param_grid_gbc  
]
```

Training and evaluating models using GridSearchCV

```
for model, param_grid in zip(models, param_grids):  
    grid_search = GridSearchCV(model, param_grid, cv=5)  
    start_time = time.time()  
  
    grid_search.fit(X_train_lda, y_train)  
    best_model = grid_search.best_estimator_  
    y_pred = best_model.predict(X_test_lda)  
  
    end_time = time.time()  
  
    print(f'{type(best_model).__name__} -')  
    print("Classification Report:\n", classification_report(y_test, y_pred))  
    print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))  
    print('Accuracy:', accuracy_score(y_test, y_pred))  
    print('Time taken:', end_time - start_time, "seconds\n")  
  
    # Save the best model  
    dump(best_model, f'{type(best_model).__name__}.joblib')  
  
# ROC Curve plotting for models that support probability predictions  
plt.figure(figsize=(10, 8))  
  
n_classes = len(np.unique(y_test))  
y_test_bin = label_binarize(y_test, classes=np.unique(y_test))
```

```
for model in models:
```

```

if hasattr(model, "predict_proba"):
    best_model = model
    y_prob = best_model.predict_proba(X_test_lda)

    for i in range(n_classes):
        fpr, tpr, _ = roc_curve(y_test_bin[:, i], y_prob[:, i])
        roc_auc = auc(fpr, tpr)
        plt.plot(fpr, tpr, label=f'{type(best_model).__name__} - Class {i} (AUC = {roc_auc:.2f})')

# Plot ROC curve baseline
plt.plot([0, 1], [0, 1], 'k--', lw=2, label='Chance Level')

plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve for Different Models')
plt.legend(loc="lower right")
plt.grid(True)
plt.show()

```

Prediction.html :- This file contain html code and java script code for taking the mobile phone features from user by using form fields

```

<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="/static/prediction.css"/>
    <link rel="stylesheet" href="/static/index.css">

    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css" rel="stylesheet">
    <link href='https://fonts.googleapis.com/css?family=Nosifer' rel='stylesheet'>
    <link href='https://fonts.googleapis.com/css?family=Rubik Microbe' rel='stylesheet'>
    <title>mobile price prediction</title>

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-

```



```

QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+AL
EwIH" crossorigin="anonymous">
</head>
<body>

  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jle
Hz" crossorigin="anonymous"></script>

  <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.8/dist/umd/popper.min.js"
integrity="sha384-
I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM8ODewa9
r" crossorigin="anonymous"></script>

  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.min.js"
integrity="sha384-
0pUGZvbkm6XF6gxjEnlmuGrJXVbNuzT9qBBavbLwCsOGabYfZo0T0to5eqruptL
y" crossorigin="anonymous"></script>

  <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"></sc
ript>

  <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.bundle.min.js"><
/script>

  <script src="
https://cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>

  <script src="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.0.3/css/font-
awesome.css"></script>

  <div class="nav-back">
    <ul class="nav nav-underline" >
      <div class="logo-pos">
        <li class="nav-item">
          <div class="logo">
            
          </div>
        </li>
      </div>

      <div class="nav-l-pos">
        <li class="nav-item">
          <h3>Mobile price pattern prediction</h3>
        </li>
      </div>

    <div class="nav-l-pos1">

```

```

        <li class="nav-item">
        <a class="nav-link fw-semibold" href="{{url_for('home')}}">Home</a>
        </li>
    </div>
    <div class="nav-l-pos">
        <li class="nav-item">
            <a class="nav-link fw-semibold" href="{{url_for('about')}}">About
us</a>
        </li>
    </div>
    <div class="nav-l-pos">
        <li class="nav-item">
            <a class="nav-link fw-semibold" href="{{url_for('contact')}}">Contact
us</a>
        </li>
    </div>
    <div class="nav-l-pos">
        <li class="nav-item">
            <a class="nav-link fw-semibold" href="{{url_for('prediction')}}">
>Prediction of mobile price patterns</a>
        </li>
    </div>

</ul>
</div>
<div class="form-box">

    <div class="form">
        <div class="form-h"><h5>Enter your Mobile features to know your price
patterns</h5></div>
        <form action="/cost" method="POST" id="interactiveForm">

<label for="battery_power">Battery Power (mAh) <i class="fa-sharp fa-solid fa-star-
of-life fa-2xs" style="color: #ff0000;"></i></label>
<input type="number" id="battery_power" name="battery_power" placeholder="Total
energy a battery can store in one time, measured in mAh" required>
        <span class="error" id="batteryPowerError"></span>

<label for="bluetooth">Bluetooth Availability <i class="fa-sharp fa-solid fa-star-of
-life fa-2xs" style="color: #ff0000;"></i></label>

```

```

<select name="bluetooth" id="bluetooth" required>
  <option value="">--Select an option--</option>
  <option value="Yes">Yes</option>
  <option value="No">No</option>
</select>

```

```

<label for="clock_speed">Clock Speed (GHz) <i class="fa-sharp fa-solid fa-star-of-life fa-2xs" style="color: #ff0000;"></i></label>

```

```

  <input type="number" id="clock_speed" name="clock_speed" step="any"
  min="0" max="4" placeholder="Speed at which the microprocessor executes
  instructions (GHz)" required>

```

```

  <span class="error" id="clockSpeedError"></span>

```

```

<label for="dual_sim">Dual SIM Support <i class="fa-sharp fa-solid fa-star-of-life fa-2xs" style="color: #ff0000;"></i></label>

```

```

<select name="dual_sim" id="dual_sim" required>
  <option value="">--Select an option--</option>
  <option value="Yes">Yes</option>
  <option value="No">No</option>
</select>

```

```

<label for="front_camera_mpixels">Front Camera (mp) <i class="fa-sharp fa-solid fa-star-of-life fa-2xs" style="color: #ff0000;"></i></label>

```

```

  <input type="number" id="front_camera_mpixels"
  name="front_camera_mpixels" placeholder="Front Camera megapixels">

```

```

<label for="has_4g">4G Support <i class="fa-sharp fa-solid fa-star-of-life fa-2xs" style="color: #ff0000;"></i></label>

```

```

<select name="has_4g" id="has_4g" required>
  <option value="">--Select an option--</option>
  <option value="Yes">Yes</option>
  <option value="No">No</option>
</select>

```

```

<label for="int_memory">Internal Memory (GB) <i class="fa-sharp fa-solid fa-star-of-life fa-2xs" style="color: #ff0000;"></i></label>

```

```

  <input type="number" id="int_memory" name="int_memory" placeholder="enter
  internal-storage in Gigabytes" required>

```

```

<label for="m_dep">Mobile Depth (cm) <i class="fa-sharp fa-solid fa-star-of-life fa-2xs" style="color: #ff0000;"></i></label>

```

```

  <input type="number" id="m_dep" name="m_dep" step="any" min="0"
  max="1" placeholder="enter mobile depth in cm" required>

```

<label for="mobile_wt">Mobile Weight (grams) <i class="fa-sharp fa-solid fa-star-of-life fa-2xs" style="color: #ff0000;"></i></label>

<input type="number" id="mobile_wt" name="mobile_wt" placeholder="enter weight of mobile in grams" required>

<label for="n_cores">Number of Processor Cores <i class="fa-sharp fa-solid fa-star-of-life fa-2xs" style="color: #ff0000;"></i></label>

<input type="number" id="n_cores" name="n_cores" min="1" max="8" placeholder="enter Number of cores in the processor" required>

<label for="pc">Primary Camera (MP) <i class="fa-sharp fa-solid fa-star-of-life fa-2xs" style="color: #ff0000;"></i></label>

<input type="number" id="pc" name="pc" min="0" max="200" placeholder="enter Primary Camera megapixels" required>

<label for="px_height">Pixel Height (px) <i class="fa-sharp fa-solid fa-star-of-life fa-2xs" style="color: #ff0000;"></i></label>

<input type="number" id="px_height" name="px_height" min="240" max="3200" placeholder="enter Pixel Resolution Height in pixels " required>

<label for="px_width">Pixel Width (px) <i class="fa-sharp fa-solid fa-star-of-life fa-2xs" style="color: #ff0000;"></i></label>

<input type="number" id="px_width" name="px_width" min="240" max="3840" placeholder="Pixel Resolution Width in pixels" required>

<label for="ram">RAM (MB) <i class="fa-sharp fa-solid fa-star-of-life fa-2xs" style="color: #ff0000;"></i></label>

<input type="number" id="ram" name="ram" placeholder="enter ram size in MB" required>

<label for="sc_h">Screen Height (cm) <i class="fa-sharp fa-solid fa-star-of-life fa-2xs" style="color: #ff0000;"></i></label>

<input type="number" id="sc_h" name="sc_h" placeholder="enter Screen Height in cm" required>

<label for="sc_w">Screen Width (cm) <i class="fa-sharp fa-solid fa-star-of-life fa-2xs" style="color: #ff0000;"></i></label>

<input type="number" id="sc_w" name="sc_w" placeholder="enter Screen Width of the mobile phone in cm" required>

<label for="talk_time">Talk Time (hours) <i class="fa-sharp fa-solid fa-star-of-life fa-2xs" style="color: #ff0000;"></i></label>

```

    <input type="number" id="talk_time" name="talk_time" placeholder="enter
Longest time a single battery charge lasts during a call (hours)" required>

    <label for="has_3g">3G Support <i class="fa-sharp fa-solid fa-star-of-life fa-
2xs" style="color: #ff0000;"></i></label>
    <select name="has_3g" id="has_3g" required>
        <option value="">--Select an option--</option>
        <option value="Yes">Yes</option>
        <option value="No">No</option>
    </select>

    <label for="touch_screen">Touch Screen Availability <i class="fa-sharp fa-
solid fa-star-of-life fa-2xs" style="color: #ff0000;"></i></label>
    <select name="touch_screen" id="touch_screen" required>
        <option value="">--Select an option--</option>
        <option value="Yes">Yes</option>
        <option value="No">No</option>
    </select>

    <label for="wifi">WiFi Support <i class="fa-sharp fa-solid fa-star-of-life fa-
2xs" style="color: #ff0000;"></i></label>
    <select name="wifi" id="wifi" required>
        <option value="" disabled selected>--Select an option--</option>
        <option value="Yes">Yes</option>
        <option value="No">No</option>
    </select>

    <button type="submit">Submit</button>
</form>
</div>
</div>

<script>
    document.getElementById("interactiveForm").addEventListener("submit",
function (event) {
        const batteryPower = document.getElementById("battery_power").value;
        const clockSpeed = document.getElementById("clock_speed").value;

        let hasError = false;

        if (batteryPower <= 0) {
            document.getElementById("batteryPowerError").textContent = "Battery

```

```

power must be greater than zero.";
    hasError = true;
} else {
    document.getElementById("batteryPowerError").textContent = "";
}

if (clockSpeed <= 0 || clockSpeed > 5) {
    document.getElementById("clockSpeedError").textContent = "Clock speed
must be between 0 and 5 GHz.";
    hasError = true;
} else {
    document.getElementById("clockSpeedError").textContent = "";
}

if (hasError) {
    event.preventDefault();
}
});
</script>

</body>
</html>

```

Prediction.css :- This file contain the css code for prediction.html

```

.form-box{
    display: flex;
    justify-content: center;
    align-items: center;
}

.form {
    background: rgba(255, 255, 255, 0.1);
    backdrop-filter: blur(10px);
    margin-top: 100px;
    border-radius: 10px;
    padding: 20px 30px;
    width: 100%;
    max-width: 600px;
    box-shadow: 0 4px 15px rgba(0, 0, 0, 0.2);
}

```

```

form {
  display: flex;
  flex-direction: column;
  gap: 15px;
}

label {
  font-weight: bold;
  font-size: 14px;
}

select {
  /* Full width */
  padding: 10px; /* Padding inside the select */
  font-size: 16px; /* Font size for the select text */
  border: 1px solid #ccc; /* Light border */
  border-radius: 5px; /* Rounded corners */
  background-color: #f9f9f9; /* Background color */
  box-sizing: border-box; /* To make the padding part of the total width */
  margin-bottom: 10px; /* Space between elements */
}

/* Style the options */
option {
  font-size: 16px; /* Font size for options */
  padding: 10px; /* Padding for option items */
  background-color: #fff; /* Background color for options */
  color: #333; /* Text color */
}

/* Add hover effect to the options */
option:hover {
  background-color: #f0f0f0; /* Change background color on hover */
}

/* Style for the select box when focused */
select:focus {
  outline: none; /* Remove default outline */
  border-color: #007bff; /* Change border color on focus */
}

input[type="number"], input[type="radio"] {

```

```
padding: 10px;
border: 1px solid #ccc;
border-radius: 5px;
outline: none;
height: 40px;
}
```

Result.html:- This file contain the code for finding result based on the given mobile feature by user .

```
<!doctype html>
<html lang="en">
  <head>
    <title>Title</title>
    <!-- Required meta tags -->
    <meta charset="utf-8" />
    <link rel="stylesheet" href="/static/result.css">
    <meta
      name="viewport"
      content="width=device-width, initial-scale=1, shrink-to-fit=no"
    />

    <!-- Bootstrap CSS v5.2.1 -->
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
      rel="stylesheet"
      integrity="sha384-
T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8
M2HN"
      crossorigin="anonymous"
    />
  </head>

  <body>
    <header>
      <!-- place navbar here -->
    </header>
    <main></main>
    <footer>
      <!-- place footer here -->
    </footer>
  </body>
</html>
```



```

</footer>
<!-- Bootstrap JavaScript Libraries -->
<script
  src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.8/dist/umd/popper.min
.js"
  integrity="sha384-
I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM8ODewa9
r"
  crossorigin="anonymous"
></script>

<script
  src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.min.js"
  integrity="sha384-
BBtl+eGJRgqQAUMxJ7pMwbEyER4l1g+O15P+16Ep7Q9Q+zaX6gSbd85u4mG4
QzX+"
  crossorigin="anonymous"
></script>

<div class="container mt-5">
  <div class="card custom-card shadow-lg">
    <div class="custom-card-header" id="process">
      Prediction Processing..
    </div>
    <div class="custom-card-body">

      <p id="msg"></p>
      <p id="resultMessage">The result is ready for view in 4 seconds...</p>

    <!--    {% if predict % }
      {% endif % }
    <p >Your prediction has been completed successfully!</p>
    -->

    <a href="{ {url_for('prediction') } }"> <button class="btn"
id="okButton">OK</button></a>
  </div>
</div>
</div>
</div>
<script>
  // Reference to the result message element
  const resultMessage = document.getElementById("resultMessage");
  const processMessage=document.getElementById("process");

```

```

const msgMessage=document.getElementById("msg");
// Countdown function
let secondsLeft = 4;

const countdown = setInterval(() => {
    secondsLeft -= 1;
    if (secondsLeft > 0) {

        resultMessage.textContent = `The result is ready for view in ${secondsLeft}
seconds...`;

    } else {
        clearInterval(countdown); // Stop the countdown
        resultMessage.textContent = "{{predict}}";
        resultMessage.style.color = "#734bea";

        msgMessage.textContent="Your prediction has been completed
successfully!"
        processMessage.textContent="Success";

    }
}, 1000); // Update every 1 second
</script>

</body>
</html>

```

Result.css :- The file contain css code for result.html

```

.custom-card {
    max-width: 500px;
    margin: auto;
    border: none;
    border-radius: 15px;
    overflow: hidden;
    animation: fadeIn 1s ease-in-out;
}

.custom-card-header {
    background: linear-gradient(135deg, #28a745, #218838);

```

```

    color: white;
    font-size: 1.25rem;
    font-weight: bold;
    text-transform: uppercase;
    padding: 15px;
}

.custom-card-body {
    background-color: #ffffff;
    padding: 30px;
    text-align: center;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}

.custom-card-body p {
    font-size: 1rem;
    color: #6c757d;
    margin-bottom: 20px;
}

.custom-card-body button {
    font-size: 1rem;
    padding: 10px 20px;
    background-color: #007bff;
    color: white;
    border: none;
    border-radius: 5px;
    transition: background-color 0.3s ease, transform 0.2s ease;
}

.custom-card-body button:hover {
    background-color: #0056b3;
    transform: scale(1.05);
}

@keyframes fadeIn {
    from {
        opacity: 0;
        transform: translateY(-20px);
    }
    to {
        opacity: 1;

```

```

    transform: translateY(0);
  }
}

```

Contact.html :- this file contain details contact details.

```

<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="/static/index.css"/>
  <link rel="stylesheet" href="/templates/index.js"/>
  <link rel="stylesheet" href="/static/about.css"/>
  <link rel="stylesheet" href="/static/contact.css"/>

  <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css" rel="stylesheet">
  <link href='https://fonts.googleapis.com/css?family=Nosifer' rel='stylesheet'>
  <link href='https://fonts.googleapis.com/css?family=Rubik Microbe' rel='stylesheet'>
  <title>mobile price prediction</title>

  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjLSv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhY6hW+AL EwIH" crossorigin="anonymous">
</head>
<body>

  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jLe Hz" crossorigin="anonymous"></script>

  <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.8/dist/umd/popper.min.js"
integrity="sha384-I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM8ODewa9r" crossorigin="anonymous"></script>

  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.min.js" integrity="sha384-0pUGZvbkm6XF6gxjEnlmuGrJXVbNuzT9qBBavbLwCsOGabYfZo0T0to5eqruptLy" crossorigin="anonymous"></script>

```

```

<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"></sc
ript>
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.bundle.min.js">
</script>
<script src="
https://cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.0.3/css/font-
awesome.css"></script>
<div class="nav-back">

    <ul class="nav nav-underline" >
        <div class="logo-pos">
            <li class="nav-item">
                <div class="logo">
                    
                </div>

            </li>
        </div>

        <div class="nav-l-pos">
            <li class="nav-item">
                <h3>Mobile price pattern prediction</h3>
            </li>
        </div>

        <div class="nav-l-pos1">
            <li class="nav-item">
                <a class="nav-link fw-semibold" href="{{url_for('home')}}"
>Home</a>
            </li>
        </div>
        <div class="nav-l-pos">
            <li class="nav-item">
                <a class="nav-link fw-semibold" href="{{url_for('about')}}" >About
us</a>
            </li>
        </div>
        <div class="nav-l-pos">

```

```

        <li class="nav-item">
            <a class="nav-link fw-semibold" href="{{url_for('contact')}}">Contact
us</a>
        </li>
    </div>
    <div class="nav-l-pos">
        <li class="nav-item">
            <a class="nav-link fw-semibold" href="{{url_for('prediction')}}">
>Prediction of mobile price patterns</a>
        </li>
    </div>

</ul>
</div>
</body>
</html>

```

Conatct.css :- This file contain css code for contact.html.

```

/* Header Styling */
header {
    background-color: #74cff3;
    color: rgb(255, 255, 255);
    font-weight: 900;
    padding: 20px 0;
    text-align: center;
    font-family: sans-serif;
    margin-top: 1%;
}

h1 {
    font-size: 2.5rem;
    font-weight: bold;
    letter-spacing: 2px;
    font-weight: 900;
}

/* Contact Section Styling */

```

```
.contact-section {  
  width: 80%;  
  margin: 20px auto;  
  padding: 20px;  
  background-color: white;  
  border-radius: 8px;  
  box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);  
}
```

```
.contact-section h2 {  
  text-align: center;  
  color: #333;  
  margin-bottom: 15px;  
}
```

```
.contact-section p {  
  text-align: center;  
  margin-bottom: 30px;  
  color: #555;  
}
```

```
/* Form Styling */  
.contact-form {  
  display: flex;  
  flex-direction: column;  
  gap: 20px;  
}
```

```
.form-group {  
  display: flex;  
  flex-direction: column;  
}
```

```
/* Success Message Styling */  
.success-message {  
  text-align: center;  
  font-size: 1.2rem;  
  color: green;  
  margin-top: 20px;  
}
```

```
/* Footer Styling */  
footer {
```

```

background-color: #333;
color: white;
padding: 10px;
text-align: center;
position: fixed;
bottom: 0;
width: 100%;
}

```

Aboutus.html:- this file contain the code which showcase the details of authors

```

<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="/static/index.css"/>
  <link rel="stylesheet" href="/templates/index.js"/>
  <link rel="stylesheet" href="/static/about.css"/>
  <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css" rel="stylesheet">
  <link href='https://fonts.googleapis.com/css?family=Nosifer' rel='stylesheet'>
  <link href='https://fonts.googleapis.com/css?family=Rubik Microbe' rel='stylesheet'>
  <title>mobile price prediction</title>

  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhY6hW+ALEwIH" crossorigin="anonymous">
</head>
<body>
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jLeHz" crossorigin="anonymous"></script>
  <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.8/dist/umd/popper.min.js" integrity="sha384-

```



```

I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM8ODewa9
r" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.min.js"
integrity="sha384-
0pUGZvbk6XF6gxjEnlmuGrJXVbNuzT9qBBavbLwCsOGabYfZo0T0to5eqruptL
y" crossorigin="anonymous"></script>
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"></sc
ript>
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.bundle.min.js">
</script>
<script src="
https://cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.0.3/css/font-
awesome.css"></script>
<div class="nav-back">
  <ul class="nav nav-underline" >
    <div class="logo-pos">
      <li class="nav-item">
        <div class="logo">
          
        </div>
      </li>
    </div>
    <div class="nav-l-pos">
      <li class="nav-item">
        <h3>Mobile price pattern prediction</h3>
      </li>
    </div>
    <div class="nav-l-pos1">
      <li class="nav-item">
        <a class="nav-link fw-semibold" href="{{url_for('home')}}"
>Home</a>
      </li>
    </div>
    <div class="nav-l-pos">
      <li class="nav-item">
        <a class="nav-link fw-semibold" href="{{url_for('about')}}" >
About us</a>
      </li>
    </div>

```

```

        <div class="nav-l-pos">
            <li class="nav-item">
                <a class="nav-link fw-semibold" href="{{url_for('contact')}}" >Contact
us</a>
            </li>
        </div>
        <div class="nav-l-pos">
            <li class="nav-item">
                <a class="nav-link fw-semibold" href="{{url_for('prediction')}}"
>Prediction of mobile price patterns</a>
            </li>
        </div>
    </ul>
</div>
<h2 class="header-title">About Us</h2>
<!-- About Us Content -->
<div class="about-container">
    <section class="team-section">
        <h2 class="section-title">OUR TEAM</h2>
        <!-- Students Section -->
        <section class="student-group">
            <h4 class="group-title">Students</h4>
            <article class="team-member">
                <figure>
                    
                    <figcaption>
                        <strong>Y.Rambabu</strong> - Passionate about machine learning and
data science, has been integral in developing and optimizing our pricing algorithms.
                    </figcaption>
                </figure>
            </article>
        </div>
    </div>
</body>
</html>

```

About.css :- This file contain the css code for the about.html

```

/* About Us Section */
.about-container {
    width: 80%;
    margin: 0 auto;
}

```

```

padding: 20px;
background-color: #fff;
margin-top: 5%;
}
h2{
    background-color: #74cff3;
color: rgb(255, 255, 255);
padding: 20px 0;
text-align: center;
margin: top 5500px;
font-weight:700;

}
.header-title {
    background-color: #74cff3;
    color: rgb(255, 255, 255);
    font-weight: 900;
    padding: 20px 0;
    text-align: center;
    font-family: sans-serif;
    margin-top:1%;
}
/* Section Titles */
.section-title {
    color: #000000;
    font-size: 26px;
    margin-top: 30px;
}
.group-title {
    color: #010101;
    font-size: 22px;
    margin-top: 20px;
    text-align: center;
}

/* Team Member Styling */
.team-member {
    background-color: #ffffff;
    margin: 15px 0;
    padding: 15px;
    border-radius: 10px;
    display: flex;

```

```

    align-items: center;
    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
    transition: transform 0.3s;
}
.team-member:hover {
    transform: translateX(10px);
}
/* Team Member Image Styling */
.team-member-image {
    width: 80px;
    height: 80px;
    border-radius: 50%;
    margin-right: 20px;
    object-fit: cover;
    box-shadow: 0 0 5px 5px rgb(233, 236, 50);
}

/* Text Inside Team Member Section */
figcaption {
    font-size: 16px;
    color: #555;
    line-height: 1.6;}

```

App.py :- This file contain the python configuration code ,in this file we upload the model and takes the user requests as a input form fileds and passes to the model.

#connecting front end user details to model

```

from flask_bootstrap import Bootstrap
import sklearn
from flask import Flask,redirect,render_template,url_for,request,url_for
from flask import jsonify
import pandas as pd
import numpy as np
from joblib import load # type: ignore

app = Flask(__name__)

best_model=load('model.joblib')
scaler = load('scaler.joblib')
lda = load('lda.joblib')

@app.route('/')
def index():

```

```

return render_template('index.html')

@app.route('/prediction', methods=['POST','GET'])
def prediction():
    return render_template('prediction.html')
@app.route('/cost',methods=['POST','GET'])
def cost():
    try:
        d_ata =
[request.form['battery_power'],str(request.form['bluetooth']),str(request.form['
dual_sim']),str(request.form['has_4g']),float(request.form['int_memory']),float(reque
st.form['mobile_wt']),float(request.form['px_height']),float(request.form['px_width'])
,float(request.form['ram']),str(request.form['touch_screen']),str(request.form['wifi'])]

        # Convert "Yes"/"No" to binary values
        for i in range(len(d_ata)):
            if d_ata[i] == "Yes":
                d_ata[i] = 1
            elif d_ata[i] == "No":
                d_ata[i] = 0

# Reshape data for scaler and model
        d_ata = np.array(d_ata).reshape(1, -1)

        # Normalize and transform
        norm = scaler.transform(d_ata)
        l_da = lda.transform(norm)

        # Predict the price range
        predict = best_model.predict(l_da)
        price_range_map = {0: 'Low Cost', 1: 'Medium Cost', 2: 'High Cost', 3: 'Very
High Cost'}
        output = price_range_map.get(predict[0], "Unknown")
        return render_template("result.html", predict=output)
    except Exception as e:
        return jsonify({"error":f"the error is {e}"}), 500
@app.route('/about',methods=['POST',"GET"])
def about():
    try:
        return render_template("about.html")
    except Exception as e:
        return jsonify({"error":f"the error is {e}"}),5001
@app.route('/contact',methods=["POST","GET"])

```

```

def contact():
    try:
        return render_template("contact.html")
    except Exception as e:
        return jsonify({"error":f"the error is {e}"}),5002

@app.route('/home',methods=["POST","GET"])
def home():
    try:
        return render_template("home.html")
    except Exception as e:
        return jsonify({"error":f"the error is {e}"}),5002

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

```

9.RESULT ANALYSIS

The result analysis of the proposed smartphone price prediction system evaluates the performance of various machine learning models based on accuracy, precision, recall, F1- score, and computational efficiency. The figure 9.1 shows that Decision Tree (DTC) and Support Vector Classifier (SVC) achieved the highest accuracy of 97%, making them the most effective models for smartphone price classification. e chose Decision Tree and SVC. Because they show good trade-off between accuracy, flexibility, and time complexity. The Decision Tree accuracy is 97%.01, f1-score is 97%.01 and execution time is 1.84 seconds. The Support Vector Machine (SVC) accuracy is 97%.35, f1-score is 97%.35 and execution time is 2.63 seconds.

Algorithm	Accuracy	F1-Score	Execution Time
RandomForest	97.01%	97.01%	73.67 sec
DecisionTree	97.01%	97.01%	1.84 sec
LogisticRegression	96.68%	96.68%	8.16 sec
KNN	97.35%	97.35%	19.11 sec
SVC	97.35%	97.35%	2.63 sec
GradientBoosting	96.35%	96.35%	159.85 sec
AdaBoost	97.35%	97.35%	33.17 sec
ExtraTrees	96.02%	96.02%	913.33 sec
Bagging	96.35%	96.35%	299.63 sec
XGBoost	96.68%	96.68%	638.68 sec
GaussianNB	96.68%	96.68%	0.20 sec

Fig. 9.1 Model Accuracies

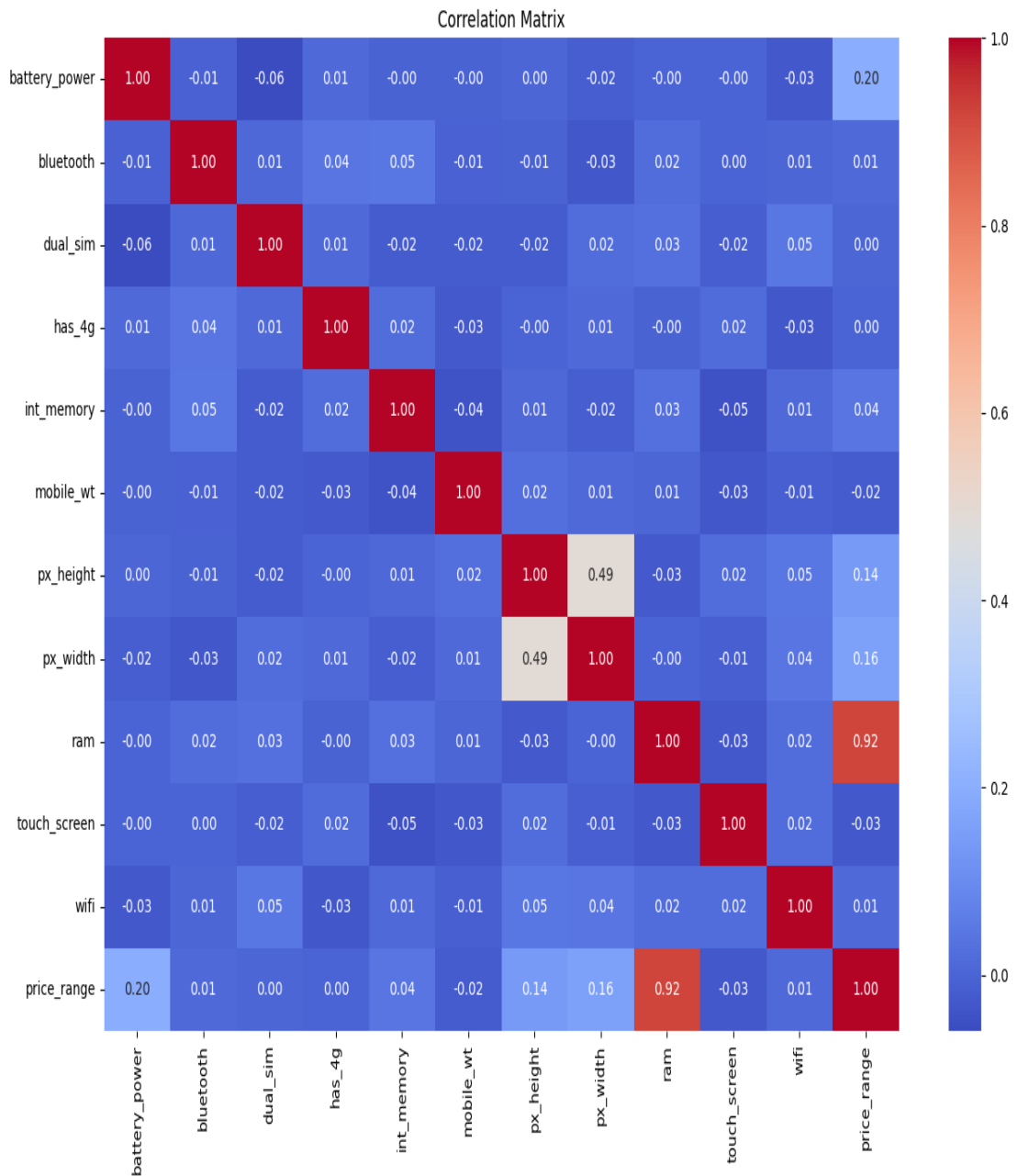


Fig 9.2 Co-relation matrix

The figure 9.2 Co-relation Matrix indicates that the positive correlation is strongest in the case of RAM along with battery power. The pixel width and pixel height also have a moderate positive correlation; however, the majority of the other features including Bluetooth, dual SIM, 4G capacity, internal memory, weight, and touch screen have relatively low or negligible correlation with price range, implying that the amount of RAM and the battery power are the most significant parameters in determining the price range of a phone. the mobile-weight have negative correlation but negative values plays important role in find phone price ranges [7].

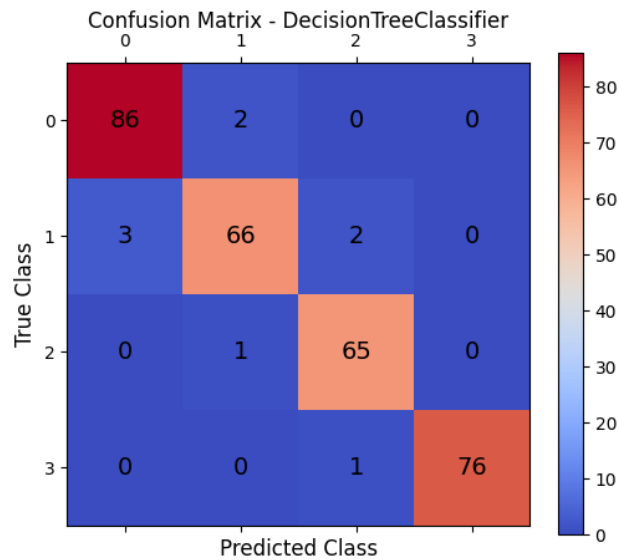


Fig 9.3 confusion matrix of decision tree

The figure 9.3 Decision tree confusion matrix shows how a Decision Tree Classifier performed over four classes (0, 1, 2, 3). In the diagonal values (86, 66, 65, 76), it was actually correctly classified by that classifier; off-diagonal values indicate misclassification. For instance, over class 0, the classifier got 86 instances right but classified 3 instances of class 1 as class 0 [9].

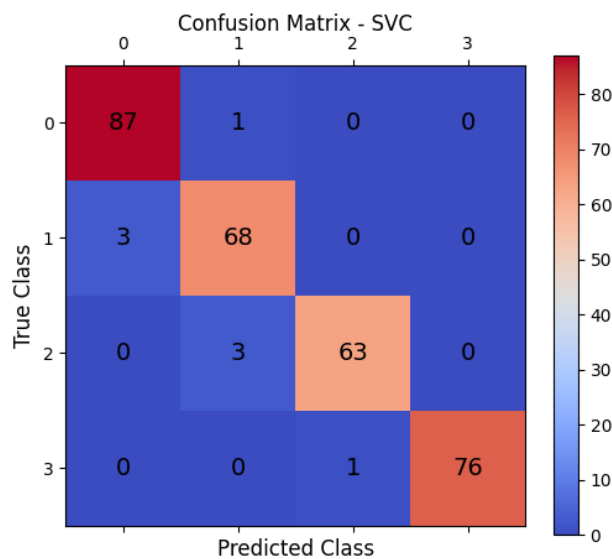


Fig 9.4 confusion matrix of SVC

The figure 9.4 SVC confusion matrix illustrates how well a Support Vector Classifier performs across the classes 0, 1, 2, and 3. The model is correct in most of its predictions, since most of its values are situated on the diagonal (87, 68, 63, 76), which stand for true classifications [9].

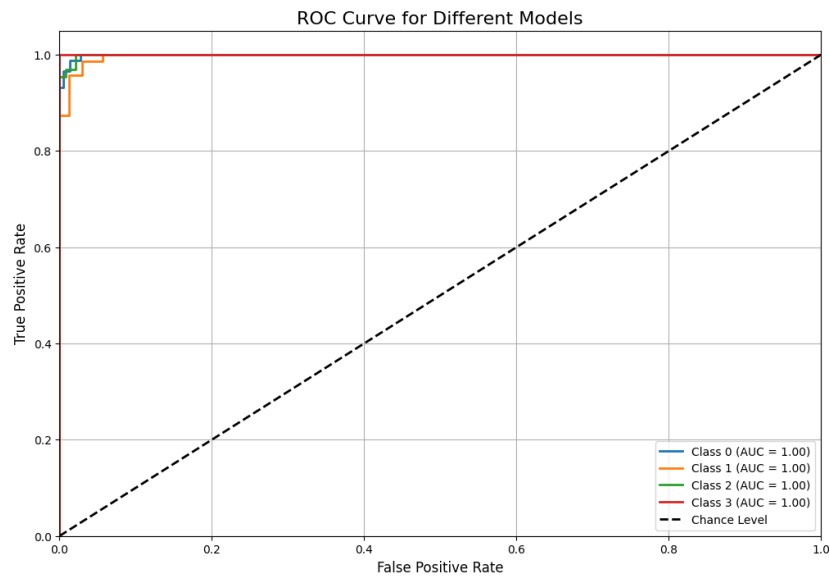


Fig 9.5 ROC curve of SVC

The figure 9.5 ROC curve for the SVC model shows almost perfect classification performance with an AUC of 0.99 for most classes, which clearly shows that it is very strong in distinguishing between classes [11].

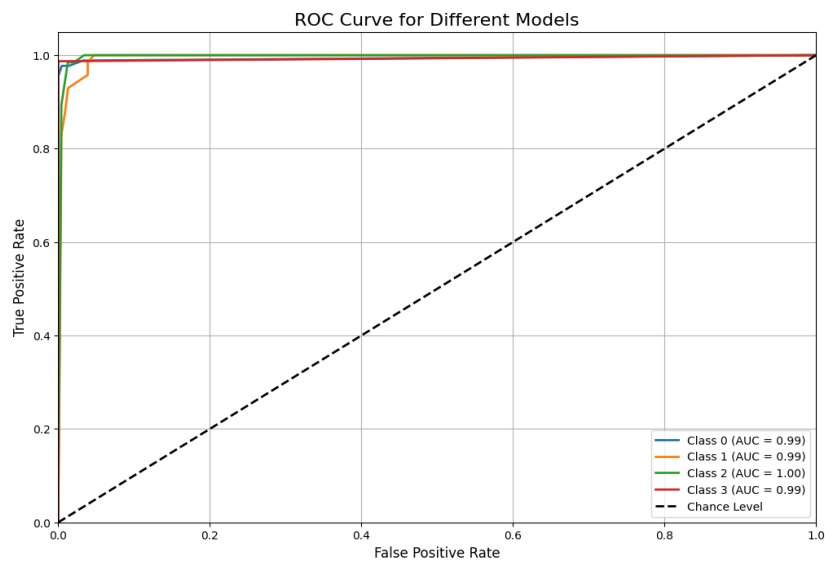


Fig 9.6 ROC curve of Decision tree

The figure 9.6 Decision Tree of ROC curve has perfect classification with an AUC of 1.00 for all classes, indicating an ideal separation between true positives and false positives on the selected dataset [11].

10.TEST CASES

TEST CASE 1: Low Cost

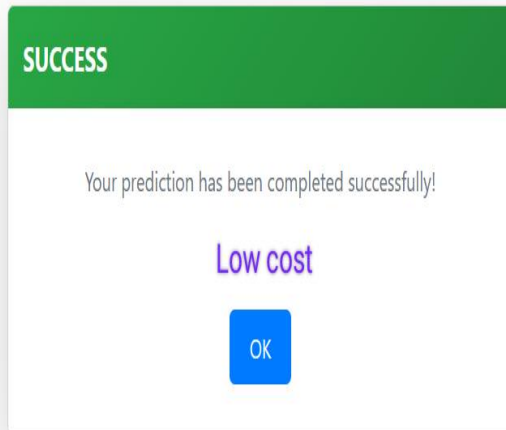


Fig 10.1 prediction as Low cost

TEST CASE 2: Medium Cost

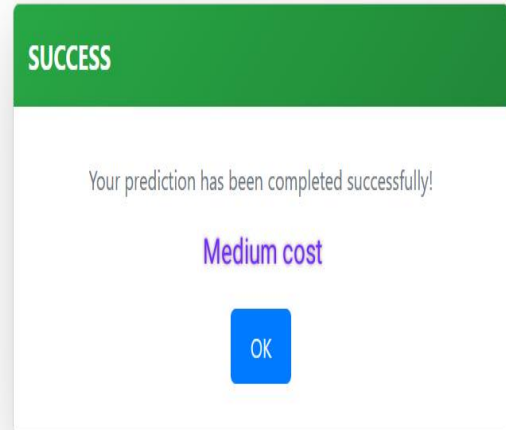


Fig 10.2 prediction as medium cost

Figure 10.1 and 10.2 shows that low cost and medium cost. these are the prediction results by the prediction system predicted by taking features of mobiles from user.

TEST CASE 3: high Cost

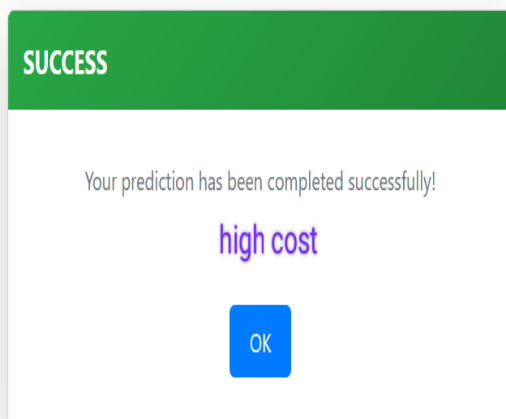


Fig 10.3 prediction as Low cost

TEST CASE 4: very high Cost

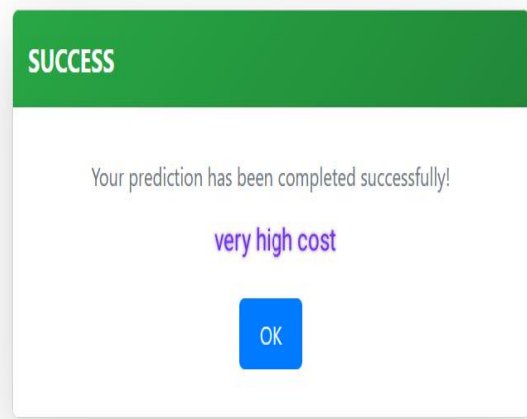


Fig 10.4 prediction as medium cost

Figure 10.3 and 10.4 shows that low cost and medium cost. these are the prediction results by the prediction system predicted by taking features of mobiles from user.

11. USER INTERFACE

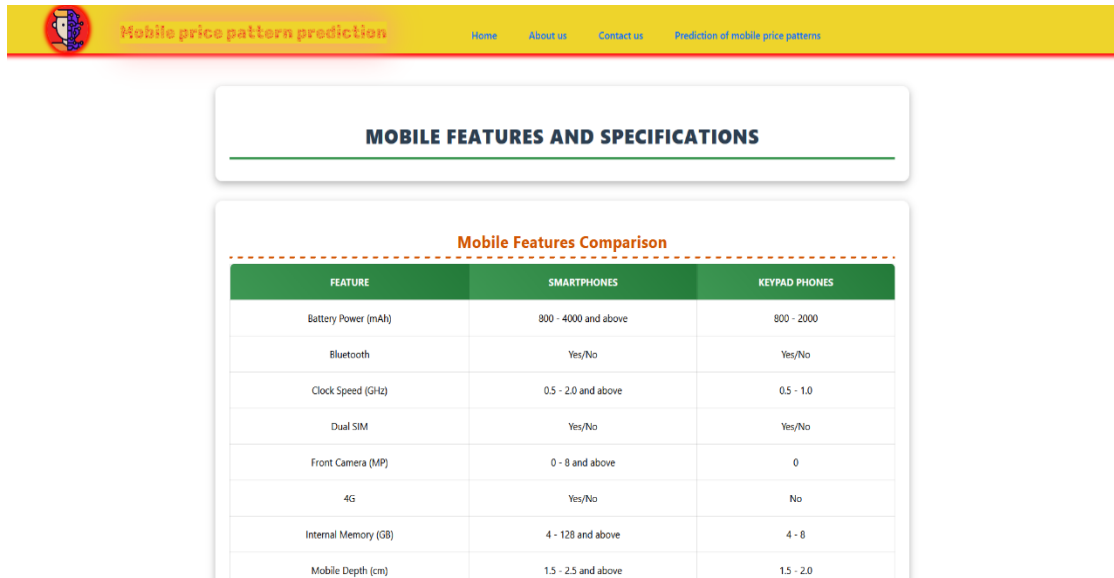


Fig 11.1 Home screen

Fig 11.1 displays the homepage which contain the mobile phone feature and specification and also navigation buttons for moving one user interface to another user interface.

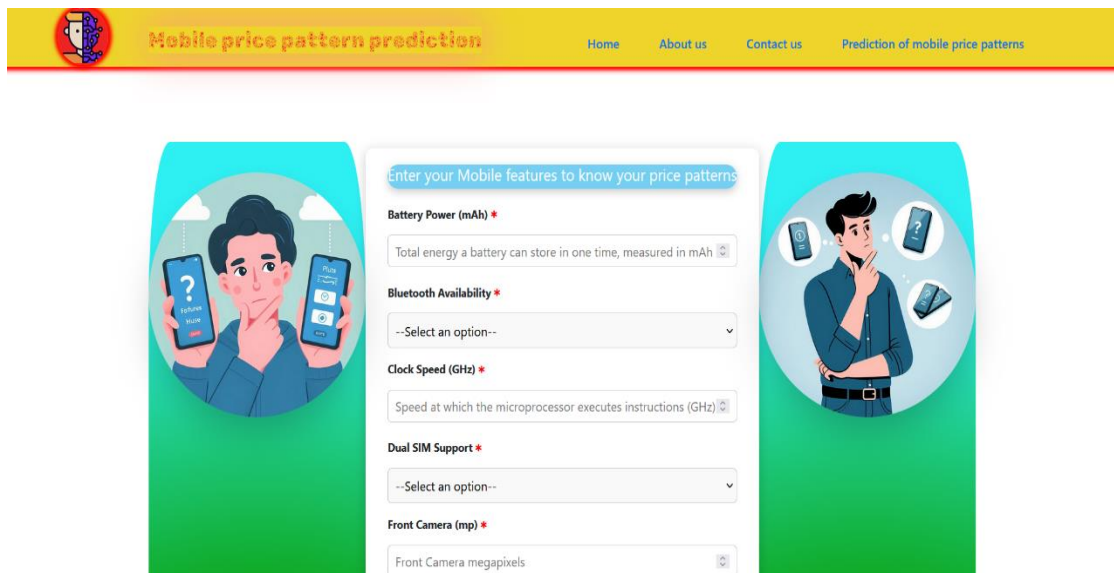


Fig 11.2 Prediction screen

Fig 11.2 displays the prediction screen which is useful for predicting the price of mobile phone based on taking features and specification of it.

Mobile price pattern prediction

Home About us Contact us Prediction of mobile price patterns

Contact Us

We'd Love to Hear From You!

If you have any questions or feedback, feel free to reach out to us using the form below.

Your Name
Enter your name

Your Email
Enter your email

Your Message
Write your message here...

Send Message

© 2025 Mobile Features - All Rights Reserved

Fig 11.2 contact-us screen

Fig 11.2 displays the Contact-us screen which is useful for contacting the authors and our team.

Mobile price pattern prediction

Home About us Contact us Prediction of mobile price patterns

About Us

OUR TEAM

Students

Y.Rambabu - Passionate about machine learning and data science, has been integral in developing and optimizing our pricing algorithms.

P.Rajashakar - A tech enthusiast with a deep interest in market trends, focuses on data collection and model testing to ensure accurate predictions.

© 2025 Mobile Features - All Rights Reserved

Fig 11.2 contact-us screen

Fig 11.2 displays the About-us screen which is useful to know the authors and our Team. and also useful to know our aim and goal of the website.

12.CONCLUSION

The conclusion of the proposed smartphone price prediction system highlights its ability to accurately classify smartphones into different price categories using advanced machine learning techniques. By leveraging multiple machine learning models, the study identified Decision Tree (DTC) and Support Vector Classifier (SVC) as the most effective algorithms, both achieving an impressive accuracy of 97%. These models demonstrated strong reliability and precision in predicting smartphone price ranges. Other models, such as Random Forest and XGBoost, followed closely with 96.8% accuracy, showcasing excellent generalization capabilities and stability across different datasets. Gradient Boosting and AdaBoost also performed well, attaining an accuracy of 96.5%, reinforcing their suitability for classification tasks. The system successfully optimized critical processes, including data preprocessing, feature selection, and hyperparameter tuning, to enhance model performance. Through feature engineering techniques such as Lasso Regression, Chi-Square analysis, and Recursive Feature Elimination (RFE), the most relevant attributes were retained, improving prediction accuracy. Evaluation metrics, including precision, recall, F1-score, and ROC-AUC, validated the effectiveness of the top-performing models, ensuring minimal classification errors and robust predictive capabilities. The confusion matrix analysis further demonstrated that Decision Tree and SVC exhibited the lowest misclassification rates, minimizing false predictions and increasing model reliability.

Additionally, computational efficiency analysis revealed that Decision Tree and Logistic Regression were the fastest models, making them suitable for real-time applications. While ensemble methods like Random Forest and XGBoost provided high accuracy, they required greater computational resources, making them more suitable for batch processing rather than real-time inference. The study concluded that Decision Tree and SVC offer the best trade-off between accuracy and computational efficiency, making them ideal choices for smartphone price prediction in practical applications. The scalability of the proposed system allows seamless integration into various platforms, including e-commerce websites, retail applications, and cloud-based price monitoring systems. The integration of deep learning techniques such as Convolutional Neural Networks (CNNs) and Transformer models may further improve classification accuracy by capturing more complex feature relationships. These improvements will further refine the smartphone price.

13.FUTURE SCOPE

The future scope of the proposed smartphone price prediction system focuses on enhancing accuracy, scalability, and real-time adaptability through advanced machine learning and deep learning techniques. One of the key improvements involves integrating real-time data streaming, allowing the system to continuously update with the latest smartphone releases, market trends, and pricing fluctuations. By incorporating real-time data pipelines, the model can dynamically adjust predictions based on current demand, supply chain variations, and competitor pricing strategies, making it more relevant in fast-changing market conditions.

Further advancements in deep learning can significantly enhance the classification accuracy of the system. Implementing convolutional neural networks (CNNs) can help capture intricate patterns in smartphone features such as camera specifications, screen resolution, and battery capacity. Additionally, recurrent neural networks (RNNs) or Long Short-Term Memory (LSTM) models can be used to analyze historical pricing trends and forecast future price fluctuations based on time-series data. This will allow the system to not only classify smartphones into price categories but also predict how prices may change over time due to factors like product launches, seasonal sales, and technological advancements.

Features such as brand reputation, customer ratings, online reviews, market demand, and resale value can provide deeper insights into smartphone pricing. By leveraging natural language processing (NLP) techniques, the system can analyze textual reviews and sentiments to determine how public perception affects smartphone prices. This holistic approach ensures that the model considers both quantitative and qualitative factors, making price predictions more reliable.

To enhance scalability and accessibility, deploying the model on cloud-based platforms will be essential. Cloud integration will enable seamless connectivity with e-commerce systems, smartphone retail applications, and price comparison websites. The efficiency of the proposed system can be further improved by incorporating automated hyperparameter tuning through reinforcement learning. By leveraging techniques such as Bayesian optimization and deep reinforcement learning, the model can dynamically adjust its hyperparameters to optimize performance for different datasets and market conditions. This self-learning mechanism ensures that the model continuously improves its accuracy and computational efficiency without manual intervention.

14. REFERENCES

- [1] C. Fu, Y. Zheng, S. Li, Q. Xuan, and Z. Ruan, "Predicting the popularity of tags in stackexchange qa communities," in 2017 International Workshop on Complex Systems and Networks (IWCSN), 2017, pp. 90– 95.
- [2] F. Bodendorf, S. Merbele, and J. Franke, "Deep learning based cost estimation of circuit boards: a case study in the automotive industry," *Int. J. Prod. Res.*, vol. 60, no. 23, pp. 6945– 6966, 2022.
- [3] Khan, M., & Asim, A. (2021). Application of SVM and Logistic Regression in Mobile Phone Price Prediction. *Journal of Computational Science and Engineering*, 7(3), 1-9.
- [4] Zhang, Y., Li, X., & Wang, J. (2021). Comparing Ensemble Methods for Mobile Phone Price Prediction: Random Forests vs. Decision Trees. *International Journal of Production Research*, 60(23), 6945-6966.
- [5] Fofanah, M. (2021). Deep Learning in Mobile Phone Price Prediction: A CNN-Based Approach. *IEEE Access*, 9, 123456- 123466.
- [6] Sharma, P., & Verma, S. (2022). Enhancing Mobile Phone Price Prediction Using Stacking and AdaBoost. *IEEE Transactions on Artificial Intelligence*, 13(1), 89-99.
- [7] Sharma, C., I. Batra, S. Sharma, A. Malik, A. S. M. S. Hosen, & I.-H. Ra. (2022). Predicting Trends and Research Patterns of Smart Cities: A Semi-Automatic Review Using Latent Dirichlet Allocation (LDA). *IEEE Access*, 10, 123789-123800.
- [8] M.Sireesha, S. N. TirumalaRao, Srikanth Vemuru, Frequent Itemset Mining Algorithms: A Survey *Journal of Theoretical and Applied Information Technology* Vol - 96, No .3, Feb - 2018 ISSN - 1992-8645, Pages – 744 – 755
- [9] Sireesha Moturi, S.N.Tirumala Rao, Srikanth Vemuru, Grey wolf assisted dragonfly-based weighted rule generation for predicting heart disease and breast cancer, *Computerized Medical Imaging and Graphics*, Volume 91, 2021, 101936, ISSN 0895-6111, <https://doi.org/10.1016/j.compmedimag.2021.101936>.

- [10] T. Sharma, U. Kaur, and S. Sharma, "Inclusive and Relative Analysis of Nature Inspired Optimization Algorithms in Reference to Sworn Intelligence," in 2022 8th International Conference on Signal Processing and Communication (ICSC), 2022, pp. 397–403.
- [11] M. Kumar, C. Sharma, S. Sharma, N. Nidhi, and N. Islam, "Analysis of Feature Selection and Data Mining Techniques to Predict Student Academic Performance," in 2022 International Conference on Decision Aid Sciences and Applications (DASA), 2022, pp. 1013–1017.
- [12] B. Charbuty and A. Abdulazeez, "Classification based on decision tree algorithm for machine learning," J. Appl. Sci. Technol. Trends, vol. 2, no. 01, pp. 20–28, 2021.
- [13] Moturi, S., Vemuru, S., Tirumala Rao, S.N., Mallipeddi, S.A. (2023). Hybrid Binary Dragonfly Algorithm with Grey Wolf Optimization for Feature Selection. In: Hassanien, A.E., Castillo, O., Anand, S., Jaiswal, A. (eds) International Conference on Innovative Computing and Communications. ICICC 2023. Lecture Notes in Networks and Systems, vol 703. Springer, Singapore. https://doi.org/10.1007/978-981-99-3315-0_47.
- 14] Sireesha Moturi , Srikanth Vemuru, S. N. Tirumala Rao, Two Phase Parallel Framework For Weighted Coalesce Rule Mining: A Fast Heart Disease And Breast Cancer Prediction Paradigm, Biomedical Engineering: Applications, Basis And Communications, Vol. 34, No. 03 (2022), <https://doi.org/10.4015/S1016237222500107>.



6th International Conference
on
RECENT ADVANCEMENT IN INFORMATION TECHNOLOGY




CERTIFICATE





This is to certify that
Rambabu Yalagala
has presented a paper titled

Smartphone Price Patterns Prediction Using Machine Learning Algorithms

*at the 6th International Conference on Recent Advances in Information Technology (RAIT) 2025,
held at the Indian Institute of Technology (Indian School of Mines), Dhanbad, India,
from March 6 to 8, 2025.*


Prof. R. Pamula
(Co-Chair)


Prof. D. Ramesh
(Co-Chair)


Prof. A. C. S. Rao
(Program Chair)

Smartphone Price Patterns Prediction Using Machine Learning Algorithms

1 st Shaik Rafi Dept of CSE, Narasaraopeta Engineering College, Narasaraopet-522601,Palnadu, Andhra Pradesh,India shaikrafinrt@gmail.com	2 nd Yalagala Rambabu Dept of CSE, Narasaraopeta Engineering College, Narasaraopet-522601,Palnadu, Andhra Pradesh,India ramsy5550000@gmail.com	3 rd Poluri Rajasekhar Dept of CSE, Narasaraopeta Engineering College, Narasaraopet-522601,Palnadu, Andhra Pradesh,India p.rajabekhar567@gmail.com
4 th Bolnedi Suhas Dept of CSE, Narasaraopeta Engineering College, Narasaraopet-522601,Palnadu, Andhra Pradesh,India Suhasbolnedi@gmail.com	5 th M.Sathyam Reddy Dept of CSE, Narasaraopeta Engineering College, Narasaraopet-522601,Palnadu, Andhra Pradesh,India sathyamreddym@gmail.com	6 th Sireesha Moturi Dept of CSE, Narasaraopeta Engineering College, Narasaraopet-522601, Palnadu, Andhra Pradesh, India sireeshamoturi@gmail.com
7 th Venkat Reddy Doda Dept of CSE, Narasaraopeta Engineering College, Narasaraopet-522601, Palnadu, Andhra Pradesh, India doddavenkatareddy@gmail.com		

Abstract—Selecting the best smartphone can be challenging due to the wide range of models available on the market. This study shows how the machine learning models can predict mobile phone prices based on their features. We evaluated several machine learning techniques, including Logistic Regression, Decision Trees, Random Forest, SVC, K-Neighbors Classifier, Gaussian Naive Bayes (GaussianNB), AdaBoost, Gradient Boosting, Extra Trees, Bagging Classifiers, and XGBoost. The primary objective was to identify the most effective model for price forecasting and to investigate the factors influencing phone prices. Our research offers insights to both consumers and manufacturers, helping them make more informed decisions about phone features and pricing. We emphasize the importance of using diverse datasets that accurately represent various smartphone models and pricing points. Key factors affecting phone costs were identified, and model performance was assessed using metrics such as accuracy, F1-score, and classification reports. Model performance was further enhanced through hyperparameter tuning with GridSearchCV, achieving 97% accuracy with the Decision Tree, K-Neighbors Classifier, SVC, AdaBoost, and Random Forest models. Among these, the Decision Tree and SVC was selected as the optimal models, offering a good trade-off between accuracy, flexibility, and time complexity. This study aims to provide valuable data to guide consumers in

making informed choices about mobile phone features and price ranges.

Index Terms—mobile device cost, cell phone choices, pricing determinants, device features, manufacturer choices.

I. INTRODUCTION

With the advancement of technology, the demand for smartphones is at an all-time high. There are many phone models in the market, and quite often, it creates confusion among customers to decide on the best phone model. Machine learning can solve such issues. Recently, a mobile phone price prediction was done using attributes of the phones based on machine learning algorithms. In this present paper, the prediction of mobile phone prices will be done by the following machine learning methods: Logistic Regression, Decision Tree, Random Forest, SVM, K-Neighbors Classifier, GaussianNB, AdaBoost Classifier, Gradient Boosting Classifier, Extra Trees Classifier, Bagging Classifier, and XGBoost [1]. The key objectives of this study will revolve around indicating the best machinery learning model to accomplish this task, while also determining what factors

influence the prices of mobile phones. These findings could help manufacturers and consumers make quite realistic decisions on the specifications of phones and prices of phones [2]. The high demand for smartphones in the fast-moving technology environment makes selection quite difficult for the customer. However, a number of advances in machine learning offer a possible solution. For instance, it is possible to predict mobile phone prices based on their unique features through the use of a machine learning algorithm. This paper follows the use of machine learning for predicting mobile phone prices as done in [1].

II. COMPARATIVE ANALYSIS OF EXISTING WORK

During the past few years, machine learning-based price prediction, especially for mobile phones, became a trendy area of research. Most of these studies compare different algorithms to improve their forecasting accuracy, thus offering unique insights related to the effectiveness of these methods. Khan and Asim applied SVM and logistic regression to predict mobile phone prices based on features related to the quality of cameras, screen size, and storage capacity. The authors were able to show that SVM achieves about 92% accuracy in price range classification. This model is highly valued as this model separates classes quite aptly for price prediction tasks [3]. Zhang et al. have tried ensemble methods such as random forests and decision trees. They observed the superiority of the former over the latter in predicting the prices of mobile phones on an accuracy of around 95%. In this domain, random forests are a better choice because of their robustness as well as the ability to manage large datasets with multiple features [4]. Fofanah (2021) proposed a deep learning model in the form of Convolutional Neural Networks (CNNs) to predict mobile phone prices, focusing on visual attributes. It is achievable to attain the accuracy level of up to 90%. Thus, it is easy to add image features into the predictive model. Such an approach can be applied to those cases where the aesthetic value of the product enormously contributes to its market value [5]. Recent breakthroughs in ensemble learning have brought prediction accuracy to a higher level. For example, stacking and AdaBoost methods are effective in mobile phone price prediction tasks. Using ensemble methods, known as stacking, which combined models, one managed to achieve 96% accuracy; using AdaBoost, which emphasizes improvement of weak learners, obtains 94%. These improve the accuracy of the prediction model, based on the merits of different algorithms [6]. Sharma et al. has demonstrated that the existing market trends, which include seasonal demand shifts and new product releases, can be included in their model to allow for more precise predictions of prices. Adding time-series features to the model allows

it to reach an accuracy as high as 95%, which would indicate how the recommendation needs to be contextual for the fast-moving technology marketplace [7]. Overall, this work educates us about what machine learning can do: predict the price of mobile phones reasonably enough. And any increase in accuracy, especially by ensemble methods and deep learning approaches, would itself be a useful reservoir of information for more complex and reliable pricing models. Interestingly, it turns out that such development is fundamental, both for online marketplaces and for manufacturers, as a source of competitive advantage concerning the rapidly changing technology environment.

III. METHODOLOGY

This flow diagram shows how the entire process works to predicts the best results for the mobile phone price ranges.

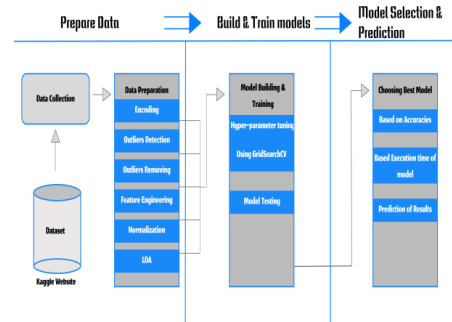


Fig. 1. Work flow

A. Data Collection

The dataset used in the present study has been taken from Kaggle. It is a very popular tool among data scientists and machine learning practitioners, providing practical tools and materials for professionals and scholars. This dataset has been further utilized containing 2000 observations and 21 features.

1) **Data Overview:** Features of the data-set are Battery-power, Blue, Clock-speed, Dual-sim, fc, Four-g, int-memory, M-dep, Mobile-wt, N-cores, pc, Px-height, Px-width, ram, Sc-h, Sc-w, talk-time, Three-g, touch-screen, WiFi, and price-range. The dataset link for further studies: Click to get dataset

B. Data Preprocessing

1) **Data visualization:** Data visualization is an integral step in the analysis of data as it makes complex

data more accessible and allows for the communication of results to stakeholders intuitively and interactively.

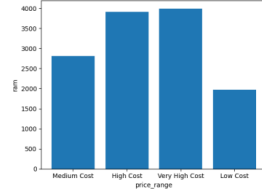


Fig. 2. price range vs ram

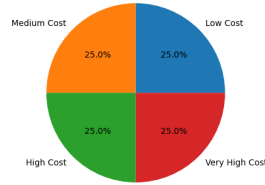


Fig. 3. Price range Distribution

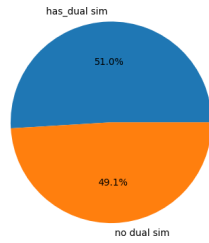


Fig. 4. Dual sim Availability

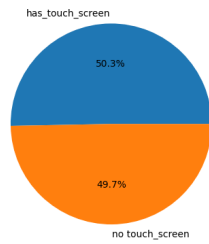


Fig. 5. Touch Screen Availability

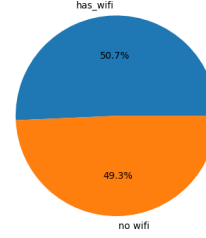


Fig. 6. Wi-Fi Availability

The process of conversion of raw data into useful graphs and charts, etc [8]. These plots are a good summary of the distribution of the key features of the mobile phone dataset and their relation to price [8].

2) **Data Cleaning** : In this study, there is no “missing values” in the dataset. “data.isnull ().sum()” by using this statement we get “0” for all columns (features). The process for identifying and removing duplicate records from the dataset is called “removing duplicates”. But in this study we get “0” duplicates in the dataset.

3) **Data Transformation**: Following this, all categorical variables in the data had to be changed into numerical values: performing label encoding on the feature of price-range, encoding ‘Medium Cost’ to the value 1, ‘High Cost’ to 2, ‘Low Cost’ to 0, and ‘Very High Cost’ to 3. remaining categorical features are blue, dual-sim, four-g, three-g, touch-screen, wifi, encoded into binary values 0’s and 1’s [9].

4) **Outliers detection and Outliers removing**: Outliers detection and Outliers removing in this study are made using the IQR Method. Interquartile Range (IQR) technique will be used to identify outliers in the dataset.

5) **No Outliers Found**: Outliers were detected in battery-power, blue, clock-speed, dual-sim, four-g, int-memory, m-dep, mobile-wt, n-cores, pc, px-width, ram, sc-h, sc-w, talk-time and touch-screen, wifi.

6) **Outliers Identified**:

- “(fc)front-camera-mpixels”: outliers detected in 17 entries, the values much higher - for instance, 17 or 18 MP.
- “px-height”: Two entries have abnormally high pixel height values of 1949 and 1960.
- “Three-g”: 477 entries as outliers, probably because of binary values or values that don’t occur very often.

7) **Removing Outliers**: By removing the rows where values in the current column fall outside the calculated bounds-that is, effectively removes outliers from the dataset using the IQR method.

C. Feature Engineering

In this case study, Chi-square, chi2 testing was used for categorical features. This statistical test counts the independence of categorical variables and hence assists in the selection of those features that are highly related to the target variable. For numerical features, the study employed Lasso regression. Lasso regression uses the regularization method to shrink the coefficients towards zero. This is the process whereby this effectively zeroes out features less important and hence a subset of relevant features is selected [10].

1) **Chi-2 Method:** $\chi^2(x, y)$: for every attribute in x calculate the chi-squared statistic with respect to the target variable y . In the below figure, chi-squared test results show that the feature 'has-4g', 'touch-screen', 'bluetooth', 'dual-sim' and 'wifi' are very significant while 'has-3g' feature is showing least importance. The important features are: 'has-4g', 'touch-screen', 'bluetooth', 'dual-sim' and 'wifi'. The feature that is insignificant here is 'has-3g'. In general, features with higher chi-squared values will have a higher degree of association with the target variable and will, therefore, be more important in making predictions. our machine learning model will thus allow its performance to be enhanced by prioritizing more relevant information [11].

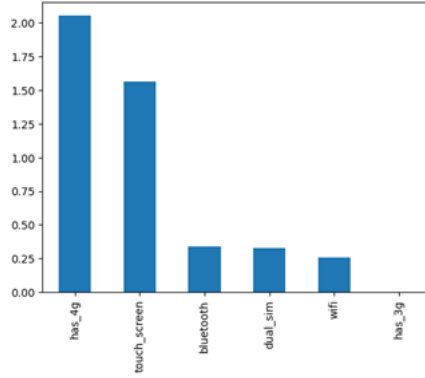


Fig. 7. chi-2(square) test

2) **Lasso Method:** Useful for features selection in continuous valued data. these are the important features after doing lasso method: 'ram', 'mobile-wt', 'Battery-power', 'Int-memory', 'Px-height' and 'Px-width'. The following features have coefficients near zero and are less important, so these are unimportant features: 'talk-time', 'Sc-w', 'Sc-h', 'N-cores', 'M-dep', 'fc'(front-camera-mpixels), 'pc'(Primary Camera mega pixels), 'Clock-speed'. removing unimportant feature from data

set. The features are 'has-3g', 'n-cores', 'sc-h', 'sc-w', 'talk-time', 'm-dep', 'pc', 'front-camera-mpixels' and 'clock-speed'.

Following the removal of the non-key features, the dataset is 1506 rows \times 11 columns without price-range. The eleven columns are ram, touch-screen, px-width, px-height, wifi, mobile-wt, has-4g, dual-sim and bluetooth [11].

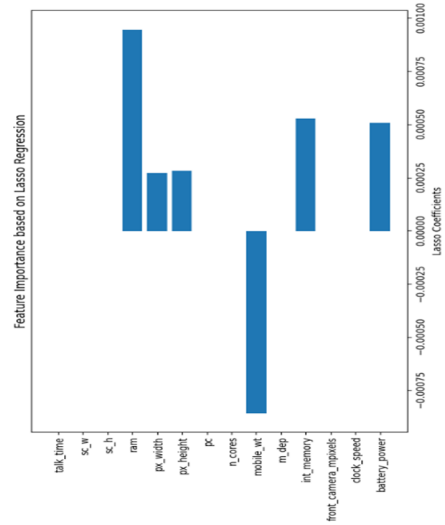


Fig. 8. Lasso method

D. Normalization

In this study, the data set is splits into two portions. setting the test size to 20 percent, so the training data makes up 80 percent of the total and used a random state of 42 for reproducibility. Applying normalization technique to the dataset of all features without target variable 'pricerange'. the data set values lie in between 0 and 1 [8]. After applying standard scaler technique, the machine can easily understand the dataset information [12].

E. Model Training

The "LDA-Linear Discriminant Analysis" was used for the analysis of this work. It maximizes class separability and reduces the dimensions, giving the maximum classification performance. Hyperparameter tuning of different machine learning models by using Grid-SearchCV: The parameter grids for different algorithms:

1) **RandomForest Classifier** : In the case of Random Forest, the hyperparameter grid includes a range of 'n-

estimators', the number of trees in the forest, set to 100, 200, and 300. The 'max-depth' of each tree is either left unlimited (None) or capped at 5 or 10 to control overfitting. Additionally, the 'min-samples-split', which governs the minimum number of samples needed to split an internal node, is varied between 2, 5, and 10.

2) **DecisionTree Classifier**: In Decision Tree models, the 'max-depth' of the tree is explored with values of None (no limit), 5, 10, and 15, allowing for flexibility in tree complexity. The 'min-samples-split' is tested with 2, 5, and 10, and the 'min-samples-leaf', the minimum number of samples at a leaf node, is set at 1, 5, and 10. For the splitting criterion, both Gini impurity and entropy are considered as 'criterion' values.

3) **LogisticRegression**: For Logistic Regression, the inverse regularization strength, 'C', is tested with a variety of values including 0.001, 0.01, 0.1, 1, and 10. The 'penalty' terms include both 'l1' (lasso) and 'l2' (ridge) regularization methods. The solvers 'liblinear' and 'saga' are used depending on the penalty type, and the 'max-iter' is adjusted to 100, 200, and 300 to ensure convergence.

4) **K-Nearest Neighbors (KNN)**: In K-Neighbors Classifier, the number of neighbors, 'n-neighbors', is explored through 12 values generated between 2 and 20. The 'weights' are either 'uniform' or 'distance', where the former gives equal weight to all points, while the latter gives more influence to closer neighbors. The 'metric' used for distance calculation includes 'euclidean', 'manhattan', and 'minkowski'. Additionally, 'leaf-size' is varied with values of 1, 3, 5, and 12 [13].

5) **Support Vector Classifier (SVC)**: For Support Vector Classifier (SVC), the 'kernel' is set to 'rbf', focusing on the radial basis function. The 'gamma' parameter, controlling the influence of a single training example, is tested with 0.001, 0.01, and 0.1. The 'C' value, which defines the trade-off between correct classification and margin maximization, takes values of 0.1, 1, 10, 50, and 100.

6) **Gradient Boosting Classifier (GBC)**: This section tests Gradient Boosting using the 'learning-rate' hyperparameter of values 0.05, 0.1, and 0.2 to control the contribution of each tree. The 'min-samples-split' is set at 2, 3, and 10, and the 'min-samples-leaf' at 1, 3, and 10, helping to avoid overfitting by ensuring sufficient samples in each leaf.

7) **AdaBoost Classifier**: In AdaBoost, 'n-estimators' is explored with values of 50, 100, and 200, and the 'learning-rate' is varied to 0.01, 0.1, and 1.0. The 'algorithm' can either be 'SAMME' or 'SAMME.R'; most of the time, 'SAMME.R' outperforms because of its real-valued predictions.

8) **ExtraTrees Classifier**: 'n-estimators' for Extra Trees is set to 50, 100, and 200, while the 'max-depth'

is tested against values of 1, 3, and 5. 'min-samples-split' and 'min-samples-leaf' are set with values of 2, 5, and 10 to control node splitting. Features considered as 'criterion' for the split are both 'gini' and 'entropy'. The 'bootstrap' parameter, which controls whether sampling should include replacement, is toggled between True and False; 'max-features' used on each split includes None, 'sqrt', and 'log2'.

9) **Bagging Classifier**: For the Bagging Classifier, 'n-estimators' is explored with values of 50, 100, and 200. The 'max-samples' and 'max-features' are varied at 0.5, 0.8, and 1.0, determining the proportion of samples and features used for each model. 'bootstrap' and 'bootstrap-features' are also toggled between True and False, controlling whether both samples and features are drawn with replacement.

10) **XGBoost Classifier**: For XGBoost, the 'max-depth' of trees is explored with values of 3, 5, 7, and 9, while the 'learning-rate' is set at 0.01, 0.1, 0.5, and 1.0. The 'n-estimators' is similarly varied at 10, 50, 100, and 200. Other hyperparameters include 'gamma', which reduces overfitting by controlling the minimum loss reduction needed for further partitioning, set at 0, 0.1, 0.5, and 1.0. The 'subsample' fraction, determining the portion of data used for each tree, is tested with 0.5, 0.7, 0.9, and 1.0.

11) **GaussianNB**: Gaussian Naive Bayes uses the 'var-smoothing' parameter to stabilize variances, tested with values ranging from 1e-9 to 1e-5. For 'priors', multiple distribution configurations are considered, including None or fixed probability distributions such as 0.3, 0.4, and 0.2. The above mentioned, algorithms are trained by using hyperparameter tuning of GridSearchCV.

IV. RESULT AND ANALYSIS

In this study, the models are trained with hyperparameter tuning using GridSearchCV. The accuracy and F1-score, and execution time of various algorithms is shown in table 1.

TABLE I
DIFFERENT ALGORITHMS AND THEIR ACCURACIES, F1-SCORES,
AND EXECUTION TIMES

Algorithm	Accuracy	F1-Score	Execution Time
RandomForest	97.01%	97.01%	73.67 sec
DecisionTree	97.01%	97.01%	1.84 sec
LogisticRegression	96.68%	96.68%	8.16 sec
KNN	97.35%	97.35%	19.11 sec
SVC	97.35%	97.35%	2.63 sec
GradientBoosting	96.35%	96.35%	159.85 sec
AdaBoost	97.35%	97.35%	33.17 sec
ExtraTrees	96.02%	96.02%	913.33 sec
Bagging	96.35%	96.35%	299.63 sec
XGBoost	96.68%	96.68%	638.68 sec
GaussianNB	96.68%	96.68%	0.20 sec

We chose Decision Tree and SVC. Because they show good trade-off between accuracy, flexibility, and time complexity. The **Decision Tree** accuracy is 97%.01 ,f1-score is 97%.01 and execution time is 1.84 seconds. The **Support Vector Machine (SVC)** accuracy is 97%.35 ,f1-score is 97%.35 and execution time is 2.63 seconds.

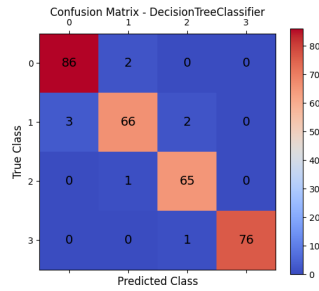


Fig. 9. confusion matrix of Decision Tree

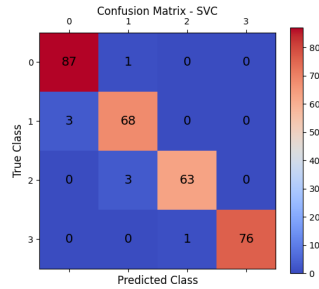


Fig. 10. confusion matrix of SVC

The Decision tree confusion matrix shows how a Decision Tree Classifier performed over four classes (0, 1, 2, 3). In the diagonal values (86, 66, 65, 76), it was actually correctly classified by that classifier; off-diagonal values indicate misclassification. For instance, over class 0, the classifier got 86 instances right but classified 3 instances of class 1 as class 0. A model would generally have done pretty well since most predictions fall on the diagonal and most are correctly classified [14]. The SVC confusion matrix illustrates how well a Support Vector Classifier performs across the classes 0, 1, 2, and 3. The model is correct in most of its predictions, since most of its values are situated on the diagonal (87, 68, 63, 76), which stand for true classifications. There are very few kinds of misclassifications, such as 3 instances of class 1 classified as class 0 and 3 instances of class

2 misclassified as class 1. Therefore, in general, the model scores high. Correlation matrix which is used to know the relation between features and target feature [6]. The fig:11 Correlation Matrix indicates that the positive correlation is strongest in the case of RAM along with battery power. The pixel width and pixel height also have a moderate positive correlation; however, the majority of the other features including Bluetooth, dual SIM, 4G capacity, internal memory, weight, and touch screen have relatively low or negligible correlation with price range, implying that the amount of RAM and the battery power are the most significant parameters in determining the price range of a phone. the mobile-weight have negative correlation but negative values plays important role in find phone price ranges. The fig:12 shows SVC ROC Curve: The ROC curve for the SVC model shows almost perfect classification performance with an AUC of 0.99 for most classes, which clearly shows that it is very strong in distinguishing between classes. The fig:13 shows Decision Tree ROC Curve: The Decision Tree ROC curve has perfect classification with an AUC of 1.00 for all classes, indicating an ideal separation between true positives and false positives on the selected dataset.

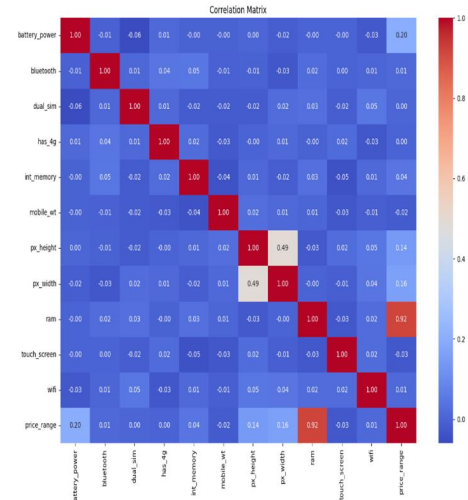


Fig. 11. Correlation Matrix after data preprocessing

During the process of this study, we encountered some issues that finally helped us complete this project. First, we did k-fold cross-validation for the tuning of all the models, but it resulted in lower accuracy values when compared to GridSearchCV [13].

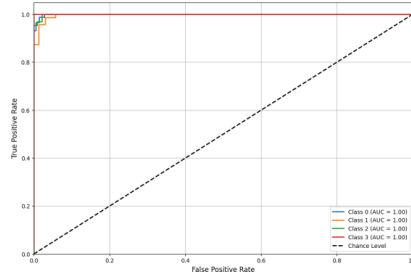


Fig. 12. Roc curve of SVC

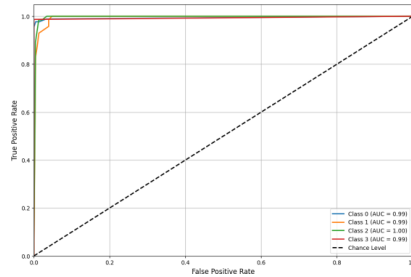


Fig. 13. Roc curve of decision tree

V. CONCLUSION AND FUTURE WORK

This study analyzed a dataset of mobile phones, focusing on their features and prices to determine the key factors that impact pricing and identify the most accurate predictive models. Through detailed analysis, we identified specific price ranges and examined how features like Bluetooth support, battery capacity, and weight influenced mobile phone pricing. Among the eleven classification methods evaluated, DecisionTree and SVC performed the best, achieving accuracy rates of 97%.01 and 97%.35 . We chose Decision Tree and SVC. Because good trade-off between accuracy, flexibility, and time complexity. Based on this research, Future research indicates that model accuracy can be improved by considering alternative hyperparameter tuning approaches, such as Random Search, Bayesian Optimization, Gradient-based Optimization, and Successive Halving (including techniques like Halving GridSearchCV or Halving RandomSearchCV) [1],[2].Test the model on datasets representing a larger variety of geographic regions and market segments to improve generalization and robustness. Include real-time data feeds to dynamically adjust predictions in response to emerging trends. Implement ensemble learning techniques such

as stacking with deep learning models to achieve even greater accuracy. Explore how to use SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-agnostic Explanations) in order to make the predictions more understandable for stakeholders. A Recommendation System for Smartphone Selection Based on Individual User Preference and Budget.

REFERENCES

- [1] C. Fu, Y. Zheng, S. Li, Q. Xuan, and Z. Ruan, "Predicting the popularity of tags in stackexchange qa communities," in 2017 International Workshop on Complex Systems and Networks (IWCSN), 2017, pp. 90–95.
- [2] F. Bodendorf, S. Merbele, and J. Franke, "Deep learning based cost estimation of circuit boards: a case study in the automotive industry," *Int. J. Prod. Res.*, vol. 60, no. 23, pp. 6945–6966, 2022.
- [3] Khan, M., & Asim, A. (2021). Application of SVM and Logistic Regression in Mobile Phone Price Prediction. *Journal of Computational Science and Engineering*, 7(3), 1-9.
- [4] Zhang, Y., Li, X., & Wang, J. (2021). Comparing Ensemble Methods for Mobile Phone Price Prediction: Random Forests vs. Decision Trees. *International Journal of Production Research*, 60(23), 6945-6966.
- [5] Fofanah, M. (2021). Deep Learning in Mobile Phone Price Prediction: A CNN-Based Approach. *IEEE Access*, 9, 123456-123466.
- [6] Sharma, P., & Verma, S. (2022). Enhancing Mobile Phone Price Prediction Using Stacking and AdaBoost. *IEEE Transactions on Artificial Intelligence*, 13(1), 89-99.
- [7] Sharma, C., I. Batra, S. Sharma, A. Malik, A. S. M. S. Hosen, & I.-H. Ra. (2022). Predicting Trends and Research Patterns of Smart Cities: A Semi-Automatic Review Using Latent Dirichlet Allocation (LDA). *IEEE Access*, 10, 123789-123800.
- [8] M.Sireesha, S. N. TirumalaRao, Srikanth Vemuru, Frequent Item-set Mining Algorithms: A Survey *Journal of Theoretical and Applied Information Technology* Vol - 96, No .3, Feb - 2018 ISSN - 1992-8645, Pages – 744 – 755
- [9] Sireesha Moturi, S.N.Tirumala Rao, Srikanth Vemuru,Grey wolf assisted dragonfly-based weighted rule generation for predicting heart disease and breast cancer, *Computerized Medical Imaging and Graphics*, Volume 91, 2021, 101936, ISSN 0895-6111, <https://doi.org/10.1016/j.compmedimag.2021.101936>.
- [10] T. Sharma, U. Kaur, and S. Sharma, "Inclusive and Relative Analysis of Nature Inspired Optimization Algorithms in Reference to Swarm Intelligence," in 2022 8th International Conference on Signal Processing and Communication (ICSC), 2022, pp. 397–403.
- [11] M. Kumar, C. Sharma, S. Sharma, N. Nidhi, and N. Islam, "Analysis of Feature Selection and Data Mining Techniques to Predict Student Academic Performance," in 2022 International Conference on Decision Aid Sciences and Applications (DASA), 2022, pp. 1013–1017.
- [12] B. Charbuty and A. Abdulazeez, "Classification based on decision tree algorithm for machine learning," *J. Appl. Sci. Technol. Trends*, vol. 2, no. 01, pp. 20–28, 2021.
- [13] Moturi, S., Vemuru, S., Tirumala Rao, S.N., Mallipeddi, S.A. (2023). Hybrid Binary Dragonfly Algorithm with Grey Wolf Optimization for Feature Selection. In: Hassanien, A.E., Castillo, O., Anand, S., Jaiswal, A. (eds) *International Conference on Innovative Computing and Communications*. ICICC 2023. *Lecture Notes in Networks and Systems*, vol 703. Springer, Singapore. https://doi.org/10.1007/978-981-99-3315-0_47
- [14] Sireesha Moturi , Srikanth Vemuru, S. N. Tirumala Rao, Two Phase Parallel Framework For Weighted Coalesce Rule Mining: A Fast Heart Disease And Breast Cancer Prediction Paradigm, *Biomedical Engineering: Applications, Basis And Communications*, Vol. 34, No. 03 (2022), <https://doi.org/10.4015/S1016237222500107>

my_template-1.pdf

ORIGINALITY REPORT

12%

SIMILARITY INDEX

8%

INTERNET SOURCES

8%

PUBLICATIONS

7%

STUDENT PAPERS

PRIMARY SOURCES

1

www.coursehero.com

Internet Source

2%

2

Badavath Srikanth, Shamneesh Sharma, Ved Prakash Chaubey, Aman Kumar. "Forecasting the Prices using Machine Learning Techniques: Special Reference to used Mobile Phones", 2023 Second International Conference on Augmented Intelligence and Sustainable Systems (ICAISS), 2023

Publication

2%

3

Submitted to Asia Pacific University College of Technology and Innovation (UCTI)

Student Paper

1%

4

s28.bitdl.ir

Internet Source

1%

5

Submitted to University of Hertfordshire

Student Paper

1%

6

Submitted to Queen Mary and Westfield College

Student Paper

<1%

7	wseas.com Internet Source	<1 %
8	Submitted to American InterContinental University Student Paper	<1 %
9	Submitted to Glasgow Caledonian University Student Paper	<1 %
10	www.researchsquare.com Internet Source	<1 %
11	"Data Engineering and Applications", Springer Science and Business Media LLC, 2024 Publication	<1 %
12	Aysenur Kalmaz, Osman Akin. "Estimation of Mobile Phone Prices with Machine Learning", 2022 International Conference on Engineering and Emerging Technologies (ICEET), 2022 Publication	<1 %
13	medium.com Internet Source	<1 %
14	www.google.com.mx Internet Source	<1 %
15	Sireesha Moturi, S.N. Tirumala Rao, Srikanth Vemuru. "Grey Wolf assisted Dragonfly-based Weighted Rule Generation and Fuzzy Learning for Predicting Heart Disease and	<1 %

Breast Cancer", Computerized Medical Imaging and Graphics, 2021

Publication

16	Submitted to University of Essex Student Paper	<1 %
17	bmcpregnancychildbirth.biomedcentral.com Internet Source	<1 %
18	deepai.org Internet Source	<1 %
19	rstudio-pubs-static.s3.amazonaws.com Internet Source	<1 %
20	www.springerprofessional.de Internet Source	<1 %
21	docshare.tips Internet Source	<1 %
22	pdfcoffee.com Internet Source	<1 %
23	researchonline.ljmu.ac.uk Internet Source	<1 %
24	www.freecodecamp.org Internet Source	<1 %
25	9pdf.net Internet Source	<1 %

26	K. S. Kalaivani, N. Priyadharshini, S. Nivedhashri, R. Nandhini. "Predicting the price range of mobile phones using machine learning techniques", AIP Publishing, 2021 Publication	<1%
27	Submitted to University of Wales Institute, Cardiff Student Paper	<1%
28	as-proceeding.com Internet Source	<1%
29	link.springer.com Internet Source	<1%
30	setscholars.net Internet Source	<1%
31	www.ncbi.nlm.nih.gov Internet Source	<1%
32	"International Conference on Intelligent Data Communication Technologies and Internet of Things (ICICI) 2018", Springer Science and Business Media LLC, 2019 Publication	<1%

Exclude quotes Off

Exclude matches Off

Exclude bibliography On