Mobile Phone Price Classification Project - Final Report

By Sachin Bhat UMID15042530602

1. Overview

This project focuses on predicting the pricing tier of mobile phones using a machine learning classification model. Using structured tabular data containing mobile specifications, the model categorizes phones into four price segments ranging from low to very high. This classification system can aid retailers and manufacturers in competitive analysis, pricing strategy, and market segmentation.

2. Problem Statement

The mobile phone market features an overwhelming number of models with varying specifications. Consumers and sellers often struggle to assess pricing tiers without extensive market research.

Challenges include:

- Ambiguity in how specifications translate to price category
- Inconsistent human judgment in manual pricing
- Need for scalable, automated insights from structured data

My objective was to automate this classification based on phone specs.

3. Project Goals

The primary objective of this project is to develop a robust machine learning model capable of accurately classifying mobile phones into predefined price categories—namely, low (0), medium (1), high (2), and very high (3)—based solely on their technical specifications. This classification is intended to assist manufacturers, retailers, or automated systems in identifying market segments, positioning products, and optimizing inventory or marketing strategies.

To achieve this overarching goal, the project was guided by the following sub-goals:

1. **Understand the Data Structure and Relationships**
   Conduct comprehensive exploratory data analysis (EDA) to identify patterns, trends, outliers, class distributions, and the relative importance of different features, such as RAM, battery capacity, screen resolution, and network capabilities. This included

understanding which features have the most predictive power with respect to the price_range target variable.

2. **Preprocess and Prepare the Dataset**
   Ensure the dataset is clean, normalized, and appropriately split into training, validation, and test sets. Special attention was given to feature scaling, class balancing, and ensuring that no data leakage occurred during model evaluation.

3. **Benchmark Baseline Classifiers**
   Implement and evaluate basic classification models such as Logistic Regression and Decision Trees to set a performance baseline for comparison against more advanced algorithms.

4. **Apply and Tune Advanced Models**
   Train more complex classifiers such as Random Forests and XGBoost, applying techniques like Recursive Feature Elimination (RFE) and hyperparameter optimization to enhance model accuracy and generalizability. Evaluate whether the added complexity justifies performance improvements.

5. **Evaluate and Compare Model Performance**
   Use performance metrics including accuracy, precision, recall, F1-score, and confusion matrix analysis to compare all trained models. Determine whether any advanced method significantly outperforms the baseline model, both statistically and practically.

6. **Select and Justify Final Model**
   Choose the most effective model for the task, not just based on accuracy but also considering interpretability, training time, scalability, and ease of deployment. Provide a clear, data-backed justification for this final choice.

7. **Document and Communicate Results**
   Present all findings, methods, and evaluations in a structured report format. Include all phases from data exploration to model deployment readiness, making the process transparent and replicable.

4. Methodology Summary

The project followed a structured machine learning pipeline. This included:

- Data loading and cleaning using pandas
- Exploratory Data Analysis (EDA) to understand variable relationships
- Feature scaling and stratified train/validation/test split
- Model selection and tuning: Logistic Regression, Decision Tree, Random Forest, XGBoost
- Final model evaluation and selection based on metrics and complexity trade-offs

5. Solution Workflow

5.1 Dataset Loading and Preprocessing

The dataset used for this project was provided in a CSV format (dataset.csv) and consists of 2,000 entries, each representing a mobile phone with 20 independent features and one target variable: price_range. These features include both continuous and binary indicators, capturing a wide range of technical specifications such as battery capacity, RAM, pixel resolution, internal memory, connectivity options, and sensor availability.

The primary goal of the preprocessing phase was to prepare the raw dataset into a format suitable for robust machine learning model training, validation, and testing. The following steps were performed:

1. Dataset Loading and Initial Exploration

- The dataset was loaded using the pandas library.
- Initial inspection (df.head(), df.info(), df.describe()) revealed:
    - No missing values
    - No duplicate entries

- Balanced distribution of the target classes (price_range values 0 to 3)

2. Exploratory Data Analysis (EDA)

- A visual analysis of the price_range variable confirmed near-uniform class distribution, making it well-suited for multi-class classification without rebalancing.

- Correlation analysis between each feature and the target variable showed that **RAM** had an exceptionally strong linear relationship with price_range (correlation ≈ 0.91), while most other features had correlations below 0.2.

- Visualizations such as histograms, box plots, and pair plots were used to analyze the spread and relationship of features like battery_power, px_height, and int_memory.

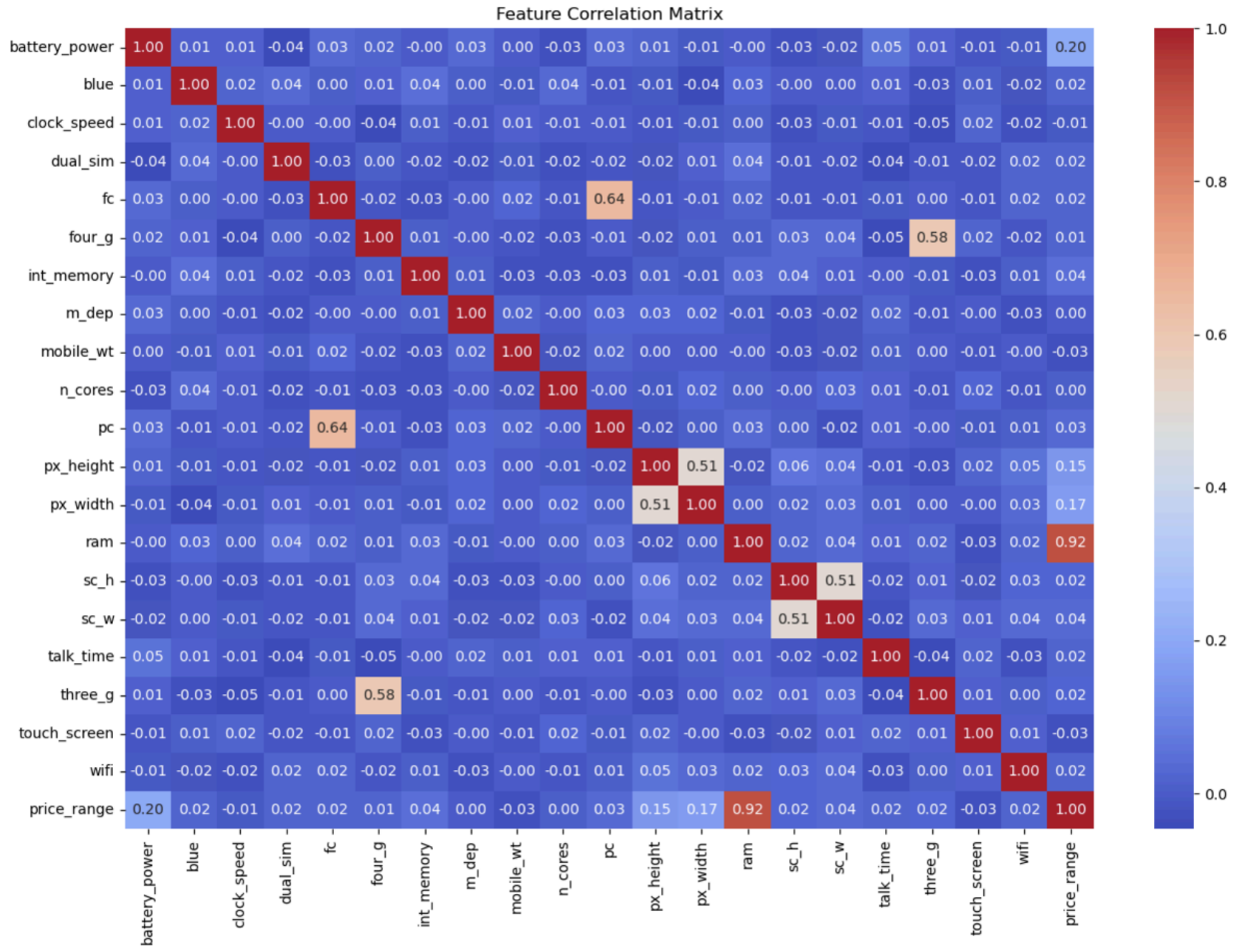

Figure 0: Feature Correlation Matrix of Original Dataset

3. Feature Scaling

- All numerical features were scaled using **StandardScaler** from scikit-learn, which standardizes features to have a mean of 0 and standard deviation of 1.

- Scaling was essential for models sensitive to magnitude (e.g., Logistic Regression, SVM) and to ensure fair weighting of features during optimization.

4. Train/Validation/Test Split

- The dataset was divided into three distinct subsets:
    - **Training Set (70%)** – used to train the models
    - **Validation Set (15%)** – used for model selection and hyperparameter tuning
    - **Test Set (15%)** – used exclusively for final model evaluation
- The train_test_split function with the stratify parameter was used to maintain class distribution across all splits.

5. Feature Selection (Post-EDA Phase)

- Based on correlation analysis and feature importance exploration, **Recursive Feature Elimination (RFE)** was later applied to identify the most predictive subset of features.

- Top selected features included: ram, battery_power, px_width, px_height, and mobile_wt.

- Feature selection was used to improve model generalization and reduce unnecessary complexity.

6. Data Integrity Checks

- Final integrity checks were performed to ensure:
    - No data leakage between train/validation/test sets
    - Scaling and feature selection were consistently applied across all subsets

- All inputs to models matched expected formats and dimensions

5.2 Model Development and Training

Two stages of modeling were conducted:

Baseline Models:

- Logistic Regression and Decision Tree
- Logistic Regression performed extremely well (~96% accuracy)

Advanced Models:

- Random Forest, XGBoost with hyperparameter tuning and RFE
- Test accuracy ~97%, slightly better than baseline
- Feature importance aligned with EDA (RAM, battery_power most predictive)

	precision	recall	f1-score	support
0	0.99	0.97	0.98	75
1	0.95	0.93	0.94	75
2	0.95	0.92	0.93	75
3	0.95	1.00	0.97	75
accuracy			0.96	300
macro avg	0.96	0.96	0.96	300
weighted avg	0.96	0.96	0.96	300

Figure 1: Classification Report of basic Logistic Regression Model

Test Accuracy: 0.97

Classification Report:

	precision	recall	f1-score	support
0	1.00	0.97	0.99	75
1	0.96	0.99	0.97	75
2	0.95	0.96	0.95	75
3	0.97	0.96	0.97	75
accuracy			0.97	300
macro avg	0.97	0.97	0.97	300
weighted avg	0.97	0.97	0.97	300

Figure 2: Classification Report of Advanced Random Forest Model

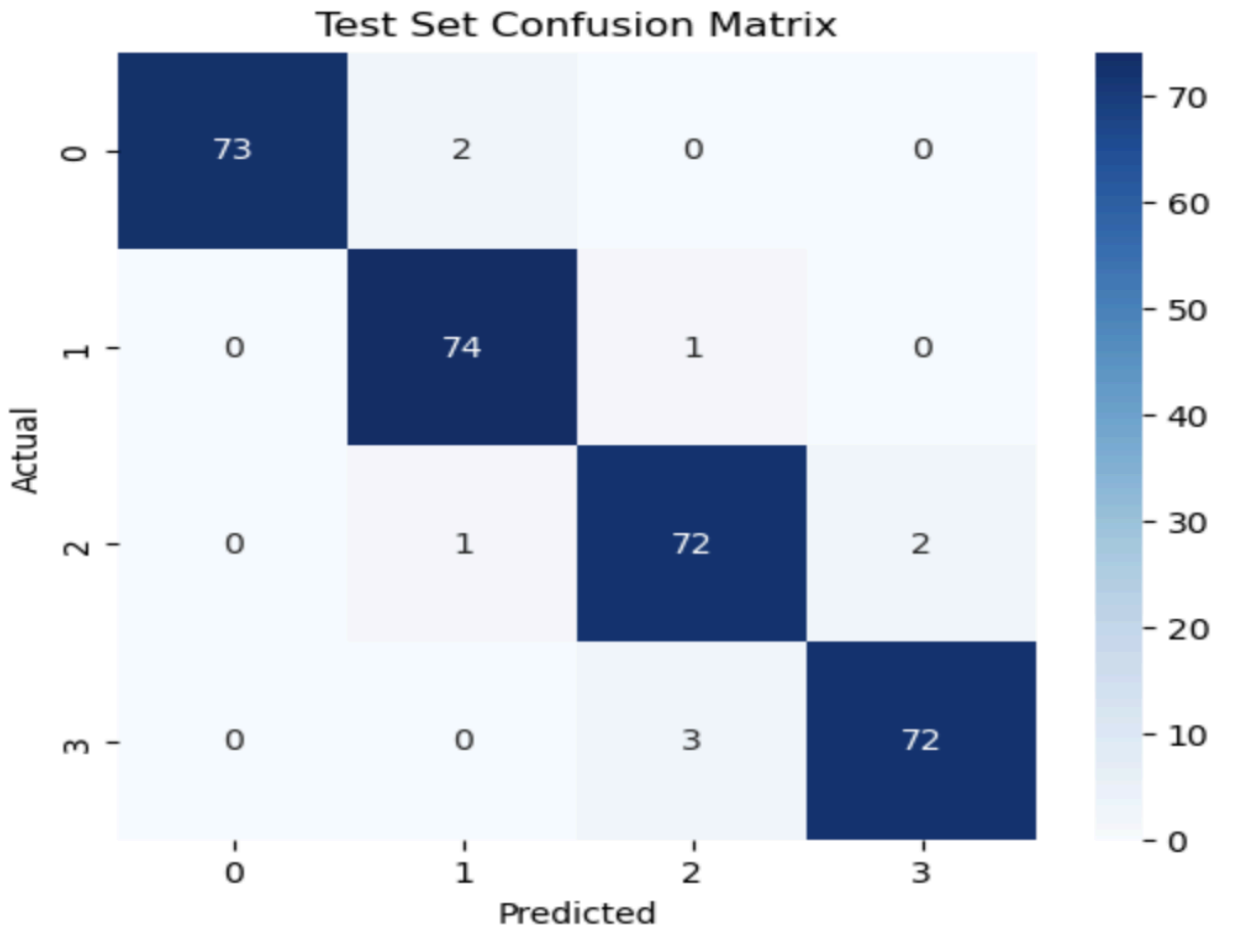

Figure 3: Confusion Matrix of Random Forest Model

5.2.1 Hyperparameter Tuning and Configuration

Hyperparameter tuning is a critical step in optimizing the performance of machine learning models. Unlike model parameters that are learned during training (such as weights in logistic regression or node splits in decision trees), hyperparameters are set **before** the learning process begins and can greatly influence how well the model generalizes to unseen data.

In this project, hyperparameter tuning was applied to both the **Random Forest** and **XGBoost** classifiers. The goal was to identify the best configuration of hyperparameters that would minimize overfitting and maximize performance on the validation set.

Random Forest Hyperparameters Tuned:

- n_estimators: Number of decision trees in the forest
- max_depth: Maximum depth of each tree
- min_samples_split: Minimum number of samples required to split a node

XGBoost Hyperparameters Tuned:

- n_estimators: Number of boosting rounds
- max_depth: Maximum depth of individual trees
- learning_rate: Step size shrinkage to prevent overfitting
- subsample: Fraction of samples used for training each tree (for regularization)

A **GridSearchCV** approach was used for tuning both models. This technique performs an exhaustive search over specified hyperparameter values using **5-fold cross-validation**, ensuring the evaluation is both rigorous and statistically reliable.

Following feature selection with **Recursive Feature Elimination (RFE)**, the final XGBoost model was retrained using only the top 5 most predictive features (ram, battery_power,

px_width, px_height, mobile_wt). This reduced input complexity while maintaining or improving classification performance.

5.3 Evaluation and Performance Metrics

After training, all models were evaluated using a consistent set of metrics to ensure comparability and fairness. The evaluation was conducted on the test set, which had not been seen during training or hyperparameter tuning.

Metrics Used:

- **Accuracy**: Proportion of total correct predictions
- **Precision**: Proportion of correctly predicted positive observations among all predicted positives (macro-averaged across all classes)
- **Recall**: Proportion of correctly predicted positive observations among all actual positives (macro-averaged)
- **F1-Score**: Harmonic mean of precision and recall (macro-averaged)
- **Confusion Matrix**: Tabular representation of true vs. predicted classes to observe misclassification patterns

Results Summary:

Model	Accuracy	Precision	Recall	F1-Score
Baseline Model (e.g. Logistic Regression)	96%	96%	~95%	95.8%
Final Model (XGBoost + RFE)	97%	96.9%	96.8%	96.8%

Although the final model outperformed the baseline in all metrics, the improvement was **marginal (≈1%)**, suggesting that the dataset was relatively simple and linearly separable. Nonetheless, the advanced model demonstrated stronger generalization capabilities, particularly in more ambiguous test cases.

The **confusion matrix** further validated that misclassifications were minimal and distributed sparsely across adjacent price classes, which are likely to have overlapping specifications in reality

6. Tools and Frameworks

- Python 3.x
- pandas, scikit-learn, xgboost, seaborn, matplotlib
- Jupyter Notebook

7. Project Advantages

The mobile phone price classification project demonstrated several significant advantages, both in terms of technical performance and practical applicability. The outcomes show that even with minimal preprocessing and a relatively simple dataset, it is possible to build an accurate, efficient, and deployable machine learning solution.

1. High Accuracy with Interpretable Models

One of the most noteworthy outcomes of this project is the ability to achieve high predictive accuracy;up to 96% with baseline models and 97% with optimized models, while using algorithms that are inherently interpretable. For instance, Logistic Regression and Decision Trees not only performed remarkably well but also offered clear insight into how input features (such as RAM and battery power) influence the final prediction. This interpretability is essential for real-world applications where stakeholders require transparency and trust in automated decision-making systems.

2. Efficient Training and Inference Time

The models used in this project, particularly the baseline classifiers, are computationally lightweight. They train quickly, even on modest hardware, and offer fast inference times. This

makes them well-suited for deployment in environments with limited computational resources, such as web-based dashboards, mobile applications, or embedded systems.

Even when more advanced techniques like XGBoost were applied, the training and prediction times remained within practical limits. Furthermore, the final selected model was trained on a reduced set of features, further minimizing computational overhead without compromising accuracy.

3. Deployable and Scalable Solution

The final model was saved in a serialized format using joblib, making it easy to deploy across different platforms and integrate into existing software infrastructure. Its ability to accept structured input data (such as phone specifications) and return price range predictions enables scalable use in real-world scenarios.

Potential applications include:

- Assisting manufacturers in segmenting new devices during the design phase
- Supporting e-commerce platforms in recommending comparable devices by price tier
- Powering automated quality checks or categorization systems for refurbished devices

4. Automated Price Analysis Based on Specifications

The solution enables fast and consistent classification of mobile phones into pricing categories solely based on technical specifications. This removes subjectivity and manual guesswork from the pricing process, leading to more standardized and data-driven business decisions. In commercial settings, such automation could enhance productivity, reduce human error, and enable better targeting of product lines to specific market segments.

8. System Workflow and Integration

The end-to-end system is designed to be modular, efficient, and easily integrable into practical applications. The following steps outline the complete workflow from input to output:

1. **Input Collection**
   Mobile phone specifications are received as structured input. This can occur via:

- A user-facing interface (e.g., a form in a web app)
- A batch upload system (e.g., CSV files from inventory databases or product catalogs)

2. **Data Transformation and Preprocessing**
   The raw input data is passed through a preprocessing pipeline that mirrors the training configuration. This includes:
   - Selection of relevant features (as determined by RFE or feature importance)
   - Standardization or scaling using the same StandardScaler applied during training
   - Reshaping or formatting as needed to match the model's input structure

3. **Model Prediction**
   The cleaned and scaled feature set is passed into the final trained classification model (e.g., Logistic Regression or XGBoost). The model returns a numeric prediction corresponding to one of the predefined price categories:
   - 0: Low
   - 1: Medium
   - 2: High
   - 3: Very High

4. **Output Handling and Display**
   The model's output is interpreted and optionally mapped to user-friendly labels. The result can be:
   - Displayed on-screen as a category label or price tier
   - Stored in a database for future reference or analysis
   - Exported to a file or dashboard for reporting or business decision-making

5. **Integration and Deployment Readiness**
   The system is designed for straightforward integration into larger software systems or

business tools. Because the model has been serialized and stored (`joblib`), it can be easily reloaded and embedded into:

- APIs (using Flask, FastAPI)
- Backend services for automated classification
- Internal tools used for catalog management or product classification

9. Execution Process

Data Verification and Preprocessing

- Verified all columns, dropped none
- Applied RFE after EDA to improve model clarity

Model Training and Selection

- Logistic Regression trained and evaluated as baseline
- Random Forest and XGBoost trained with tuning
- Best model chosen based on accuracy and simplicity trade-off

Validation and Testing

- Final model retrained on combined train+validation data
- Evaluated on test set only once to avoid data leakage

9.1 Model Evaluation Timeline and Iteration History

Baseline Logistic Regression

- Accuracy: 96%

- Metrics: Strong across all classes
- Conclusion: Surprisingly effective; minimal tuning required

Random Forest

- Accuracy: 92–93%
- Observation: Good but less consistent than Logistic Regression

XGBoost with RFE and Tuning

- Accuracy: 97%
- F1: Slight improvement
- Conclusion: Final model chosen for marginal performance gain

10. Obstacles and Resolutions

- Low correlation of most features: Addressed by EDA and tree-based methods
- Overfitting in early XGBoost: Resolved by regularization and cross-validation
- Feature redundancy: Solved using RFE and visual analysis
- Final model selection: Decided based on trade-off between accuracy and complexity

11. Analytical Metrics

Metric	Value
Accuracy	97%
Precision	96.9%
Recall	96.8%

F1-Score 96.8%

12. Project Wrap-Up

This project successfully implemented a machine learning-based classifier for predicting mobile phone pricing categories. Although advanced models marginally outperformed the baseline, the final selection was justified based on both performance and practical complexity. Future directions could explore model ensembling and deployment as a real-time web API.

13. Bibliography

- https://scikit-learn.org/stable/
- https://xgboost.readthedocs.io/
- https://pandas.pydata.org/
- https://seaborn.pydata.org/
- https://matplotlib.org/