📱 Mobile Price Classification - ML Assignment 2

Open in Streamlit
GitHub View Repository

Live App: https://2025aa05220-ml2.streamlit.app/

GitHub Source Code: https://github.com/nagarsumit/BITS-ML-Assignment-2

BITS LAB Execution BITS LAB Execution - Photo

1. Problem Statement

This project implements a machine learning pipeline to classify mobile phones into price ranges (0, 1, 2, 3) based on their technical specifications. The goal is to build an interactive Streamlit web application that allows users to compare the performance of six different classification algorithms on a hold-out test dataset.

2. Dataset Description

Source: Kaggle (Mobile Price Classification)

Dataset Name: `mobile_price_classification.csv`

Test Data: `test-data.csv` (20% hold-out set, used for evaluation)

Description:
The full dataset contains **2000 instances** and **21 features**. It involves a **Multi-class** classification problem.

- **Target Variable:** `price_range` (0: Low Cost, 1: Medium Cost, 2: High Cost, 3: Very High Cost)
- **Key Features:**
    - `ram` : Random Access Memory in Megabytes
    - `battery_power` : Total energy a battery can store in one time measured in mAh
    - `px_height` / `px_width` : Pixel Resolution Height/Width
    - `mobile_wt` : Weight of mobile phone

3. Models Used

The following six machine learning models were implemented and evaluated:

1. Logistic Regression
2. Decision Tree Classifier
3. K-Nearest Neighbor (KNN)
4. Naive Bayes (Gaussian)
5. Random Forest (Ensemble)
6. XGBoost (Ensemble)

4. Evaluation Metrics & Comparison Table

The models were evaluated on `test-data.csv`.

(Note: These values are based on the provided test split. Live app results may vary slightly depending on environment setup and scaling availability.)

ML Model Name	Accuracy	AUC Score	Precision	Recall	F1 Score	MCC Score
Logistic Regression	0.9525	0.9975	0.9529	0.9525	0.9521	0.9371
Decision Tree	0.8200	0.8798	0.8177	0.8200	0.8180	0.7606
KNN	0.5100	0.7565	0.5160	0.5100	0.5090	0.3486
Naive Bayes	0.8175	0.9544	0.8194	0.8175	0.8183	0.7567
Random Forest (Ensemble)	0.8575	0.9802	0.8555	0.8575	0.8556	0.8106
XGBoost (Ensemble)	0.9225	0.9889	0.9227	0.9225	0.9226	0.8967

5. Observations

Below are the observations on the performance of each model on the `test-data.csv` subset:

ML Model Name	Observation about model performance
Logistic Regression	**Best Performer.** Achieved the highest accuracy (~95%) and MCC score. It correctly identified that the price range is linearly dependent on features like RAM and Battery Power.
Decision Tree	Moderate performance (~82%). As expected for a single tree, it showed signs of overfitting to the training data, leading to lower generalization on the `test-data.csv` file.

ML Model Name	Observation about model performance
KNN	**Sensitive to Scaling.** Performance (~51%) was significantly lower than others. This highlights that distance-based algorithms struggle with high-dimensional, unscaled data (raw pixel specs vs RAM).
Naive Bayes	Good baseline (~81%) with a very high AUC (0.95). It proved robust and fast, effectively ranking the classes even if the absolute predicted probabilities were not perfect.
Random Forest (Ensemble)	Strong performance (~86%). It successfully improved upon the single Decision Tree by averaging multiple trees, reducing variance and providing stable predictions.
XGBoost (Ensemble)	**Top-Tier Performer.** Achieved very high accuracy (~92%) and near-perfect AUC. It handled the non-linear complexities (like screen resolution vs battery life) excellent well.

6. How to Run the App Locally

Prerequisites

- Python 3.8 or higher.
- `test-data.csv` must be present in the root directory.

Installation Steps

Step 1: Clone the repository

```
git clone [YOUR_GITHUB_REPO_LINK]
cd [YOUR_PROJECT_FOLDER]
```

Step 2: Install Dependencies

```
python -m pip install --upgrade pip
python -m pip install -r requirements.txt
```

Step 3: Run the Application

```
python -m streamlit run app.py
```