Phone Classification - Report

- Introduction -

Smartphones come with a wide range of features and prices, making it valuable to identify which specifications influence price categories. This project focuses on building machine learning models to predict the price range of a mobile phone using hardware specifications from the Phone Classification Dataset on Kaggle. The task is a **multi-class classification problem** where the target variable, price_range, includes four classes: 0 (Low cost), 1 (Medium cost), 2 (High cost), and 3 (Very high cost).
By training and evaluating various classification models, our goal is to determine which algorithms best classify phones into their respective price categories based on a selected set of numerical features. (Github model link)

- Dataset description -

The dataset provides structured, numeric information about various hardware specifications of mobile phones, with the aim of predicting their price category. Each record represents a unique phone and includes details such as RAM (memory size), battery_power (battery capacity in mAh), px_width and px_height (screen resolution in pixels), mobile_wt (device weight in grams), int_memory (internal storage capacity in GB), and talk_time (maximum talk time on a full charge, in hours). These features are all continuous numerical values, making the dataset easy to process and model. A feature importance analysis using a Random Forest model identified these seven attributes as the most predictive. The target variable, price_range, is a multi-class label ranging from 0 (low cost) to 3 (very high cost). Importantly.

- Data properties -

The data properties discussed in this section are based on the top features that were selected through feature importance analysis and used in the modeling phase:

N	ram	battery_power	px_width	px_height	mobile_wt	int_memory	talk_time
count	2000	2000	2000	2000	2000	2000	2000
mean	2124.2	1238.5	1251.5	645.1	140.2	32.0	11.0
std	1084.7	439.4	432.2	443.8	35.4	18.1	5.5
min	256	501	500	0	80	2	2
25%	1207.5	851.8	874.8	282.8	109	16	6
50%	2146.5	1226	1247	564	141	32	11
75%	3064.5	1615.3	1633	947.3	170	48	16
max	3998	1998	1998	1960	200	64	20

The summary statistics of the top 7 features used in this analysis reveal key insights into the distribution and variability of the data. The ram feature, which showed the highest importance during feature selection, ranges from 256 MB to 3998 MB, with a mean of approximately 2124 MB and a standard deviation of 1084 MB, indicating substantial variation in memory capacity among devices. Similarly, battery_power spans from 501 mAh to 1998 mAh, averaging 1238.5 mAh, reflecting a wide range of battery capacities across smartphones. Screen resolution metrics, namely px_width and px_height, also show large variability, with px_height having a minimum of 0 (possibly indicating missing or extremely low-resolution records) and a maximum of 1960 pixels. The average width and height are around 1251.5 and 645.1 pixels, respectively.

The mobile_wt feature, representing phone weight, varies between 80 g and 200 g, with most devices clustered around the mean of 140.2 g. For storage, int_memory (internal memory in GB) ranges from 2 GB to 64 GB, averaging 32 GB, which is consistent with modern mobile storage distributions. Lastly, talk_time, a battery performance indicator, ranges from 2 to 20 hours, with a mean of 11 hours, suggesting a good balance between low- and high-end devices in the dataset.

Overall, these features display diverse and realistic distributions, which contribute significantly to the model's ability to distinguish between the different smartphone price ranges.

- Data preprocessing -

The preprocessing phase began by importing the dataset from a CSV file containing mobile phone specifications. To streamline the learning process and improve model performance, we selected the **top seven most important features** identified through prior feature importance analysis using a Random Forest model: ram, battery_power, px_width, px_height, mobile_wt, int_memory, and talk_time. These features represent critical hardware aspects that directly influence the price range of smartphones.

Next, we addressed potential **missing values** by applying the SimpleImputer from scikit-learn with the **mean strategy**, which replaces any missing entries in the dataset with the average value of that column. This ensures that models trained later do not suffer from incomplete data, while also preserving the size of the dataset.

After imputation, we applied **feature scaling** using StandardScaler. This transformation standardizes all numerical features by removing the mean and scaling to unit variance. Standardization is essential because the features vary in magnitude and unit (e.g., RAM in MB vs. weight in grams), and many machine learning algorithms, especially distance-based models like KNN and margin-based models like SVM, are sensitive to these differences. Scaling ensures that each feature contributes equally to the model’s decisions.

The dataset was then split into **training and testing subsets** using a **70/30 train-test split**, with stratification based on the target variable (price_range). The choice of 70% for training provides the model with sufficient data to learn meaningful patterns, while the remaining 30% ensures a reliable and unbiased evaluation on unseen data. Stratified splitting preserves the proportion of each class in both training and test sets, which is crucial in multi-class classification tasks to avoid class imbalance bias during evaluation.

To further enhance the training set and address any class imbalance, we employed **SMOTE (Synthetic Minority Oversampling Technique)**. SMOTE generates synthetic samples for minority classes by interpolating between existing examples. This results in a more balanced training dataset, which is especially beneficial for classification tasks where some classes may be underrepresented, ensuring the model doesn't overfit to the dominant classes.

Finally, the processed datasets (X_train, X_test, y_train, y_test) were saved to CSV files for reproducibility and ease of access in subsequent modeling phases. These preprocessing steps collectively prepare the data in a clean, balanced, and standardized format that maximizes the effectiveness of machine learning algorithms.

- Model training -

To evaluate the performance of various classification techniques on the mobile price prediction task, we implemented and trained seven different machine learning models. Each model was selected for its unique learning mechanism and ability to handle multi-class classification:

- **Decision Tree Classifier**: A simple yet interpretable model that makes decisions by recursively splitting the data based on feature thresholds. In our case, we limited the tree depth to 10 (max_depth=10) to prevent it from overfitting on the training data.
- **Random Forest Classifier**: An ensemble learning method that builds a large number of decision trees and aggregates their predictions to improve accuracy and reduce overfitting. We configured it with 200 trees (n_estimators=200) for more stable and generalizable results.
- **K-Nearest Neighbors (KNN)**: A non-parametric model that predicts the class of a data point by majority vote of its closest neighbors in the feature space. We used k=5, which is a common choice for balancing bias and variance.
- **Support Vector Machine (SVM)**: A kernel-based classifier that tries to find the optimal boundary (hyperplane) between classes in a high-dimensional space. We used the RBF (Radial Basis Function) kernel with probability estimation enabled.
- **Naive Bayes**: A fast and simple probabilistic model based on Bayes' Theorem, which assumes independence among features. Although this assumption rarely holds in real datasets, Naive Bayes often performs surprisingly well for baseline comparisons.
- **Artificial Neural Network (ANN)**: A multi-layer feed-forward network built using the MLPClassifier from scikit-learn. Our network had two hidden layers with 64 and 32 neurons, respectively, and was trained for up to 1000 iterations.
- **Logistic Regression**: A linear model that estimates the probability of each class using the logistic (sigmoid) function. We increased the maximum number of iterations (max_iter=1000) to ensure proper convergence on our scaled data.

Each of these models was trained on the same preprocessed and SMOTE-balanced dataset and evaluated using standard classification metrics to ensure fair and consistent comparison.

- Model Evaluation -

Each model was trained using the SMOTE-balanced training set with scaled features, and then evaluated on the independent test set. To assess the classification performance, we used accuracy, precision, recall, and F1-score. Given the multi-class nature of our problem (price_range = 0, 1, 2, 3), we examined macro-averaged scores to equally reflect performance across all price categories.
The table below summarizes the key evaluation metrics for each model:

Model	Accuracy	Precision (1)	Recall (1)	F1-score (1)
Decision Tree	85%	85%	85%	85%
Random Forest	90%	90%	90%	90%
K-Nearest Neighbors	74%	75%	74%	74%
Support Vector Machine	92%	92%	92%	92%
Naïve Bayes	79%	78%	79%	78%
Artificial Neural Network	96%	97%	97%	97%
Logistic Regression	99%	99%	99%	99%

The performance evaluation of the classification models revealed significant variation in how well each algorithm predicted mobile phone price categories. Logistic Regression emerged as the top-performing model, achieving an outstanding 99% accuracy, along with 99% precision, recall, and F1-score. Its near-perfect metrics across all classes indicate that the data is highly linearly separable when properly scaled, and that Logistic Regression was able to fully leverage this structure. The model not only generalized well but also consistently predicted each price category with minimal misclassification, making it the most reliable and efficient choice for this task.

On the other end of the spectrum, K-Nearest Neighbors (KNN) delivered the lowest overall performance, with only 74% accuracy and relatively weaker scores in precision (75%), recall (74%), and F1-score (74%). This underperformance may be attributed to KNN's sensitivity to feature scaling, the curse of dimensionality, and the difficulty it faces when distinguishing subtle boundaries between classes in multi-class problems especially when the classes are not well-separated in feature space. Additionally, KNN relies heavily on the local structure of the data, which can be problematic in datasets like this one that require broader generalization.

Other models such as the Artificial Neural Network (ANN) and Support Vector Machine (SVM) also performed exceptionally well, with ANN achieving 96% accuracy and macro scores of 97%, reflecting its strength in capturing complex non-linear relationships. SVM followed closely with 92% across all metrics. Random Forest and Decision Tree both reached solid levels of accuracy (90% and 85%, respectively), while Naive Bayes provided a decent baseline with 79% accuracy, but struggled due to its strong independence assumptions between features.

In summary, Logistic Regression was the best because it perfectly captured the decision boundaries in the data with minimal complexity, while KNN performed the worst, likely due to its limitations in multi-class classification and its dependence on local proximity in a relatively complex feature space.

- Conclusion -

Through this project, I gained both technical and practical insights. From a technical perspective, I learned how to apply a complete machine learning pipeline from data cleaning and feature selection to model training and performance evaluation. By experimenting with multiple algorithms, I was able to clearly observe how different models behave under the same conditions, and I understood which models are better suited for structured, balanced, and numerical datasets like this one.

The most valuable outcome from the model was its ability to predict mobile phone price categories with very high accuracy especially using Logistic Regression, which achieved 99%. This gave me confidence that when the data is well-prepared and features are meaningful, even simple models can perform exceptionally well.

As for my experience working with the dataset, I developed a stronger understanding of how hardware specifications like RAM, battery life, and screen resolution affect a device's market value. This allowed me to connect machine learning output with real-world reasoning, which is something I deeply appreciated during the project.

From a real-life perspective, this project demonstrates how such models could assist companies in automated price suggestion, help customers compare devices, and even guide manufacturers in identifying which features matter most when building phones for specific price ranges. This is a direct example of how machine learning can solve real problems in commerce and product design.

Overall, this project wasn't just about building a model it was about learning how to apply data science effectively, gaining confidence in practical workflows, and appreciating how powerful and valuable data can be when used correctly.