Ramakrishna Mission Vivekananda Educational and Research Institute

Machine Learning

Final Project Report

A Classification Based approach for predicting Smartphone Price Categories

Submitted By

Sayan Das
B2430035

Raihan Uddin
B2430070

Submitted To

Br. Bhaswarachaitanya

(Tamal Maharaj)

November 25, 2024

CONTENTS

I Introduction **3**
I.1 Background 3
I.2 Motivation 4
I.3 Objectives 4

II Literature Review **6**

III Dataset Description **9**
III.1 Source 9
III.2 Features 9
III.3 Target Variable 11

IV Data Preprocessing **12**
IV.1 Data Cleaning 12
IV.1.1 Handling Missing Values 12
IV.1.2 Handling Duplicate Values 12
IV.1.3 Handling Invalid Values 12
IV.2 Outlier Handling 13
IV.3 Class Imbalance Check 14
IV.4 Correlation Analysis 15
IV.5 Feature Engineering 16

IV.6 Feature Selection . . . 17
IV.7 Normalizing features . . . 18
IV.8 Train-Test Split . . . 19
IV.9 Scaling . . . 19

V Methodology **20**
V.1 Algorithms Used . . . 20
V.2 Justification . . . 21

VI Implementation **23**
VI.1 Tools and Libraries . . . 23
VI.2 Parameters . . . 24
VI.3 Training Process . . . 25

VII Results **26**
VII.1 Plots . . . 26
VII.2 Evaluation Metrics . . . 36
VII.3 Comparison between different Models . . . 37

VIII Discussion **39**
VIII.1 Analysis of Results . . . 39
VIII.2 Anomalies and Unexpected Findings . . . 39
VIII.3 Limitations . . . 40

IX Conclusion **41**
IX.1 Key Takeaways . . . 41
IX.2 Reflection on Objectives . . . 42
IX.3 Future Work . . . 42

References **45**

SECTION I

INTRODUCTION

I.1 Background

Now a days, smartphones are more than just a means of communication. The global smartphone market is characterized by rapid technological innovation, intense competition, and increasingly sophisticated consumer expectations. With the proliferation of smartphones, the market dynamics have become increasingly complex, driven by continuous technological advancements, changing consumer preferences, and competitive pricing strategies.

The smartphone industry represents a highly dynamic technological ecosystem where manufacturers constantly strive to differentiate their products through innovative features, design, and pricing. Each season, hundreds of new smartphones are launched, each targeting different market segments and consumer needs. This rapid evolution creates significant challenges for both manufacturers and consumers in understanding and predicting smartphone pricing.

From a manufacturer's perspective, optimal pricing is crucial for maintaining market competitiveness and profitability. Pricing strategies must balance multiple factors including technological features, production costs, market positioning, and consumer purchasing power. Inaccurate pricing can lead to significant market share losses or reduced profit margins.

For consumers, purchasing a smartphone requires understanding of the complex landscape of technical specifications, brand reputation, and market trends. The ability to predict or understand the factors influencing smartphone

prices can help consumers make more informed purchasing decisions and assess the value proposition of different devices.

The emergence of machine learning techniques offers promising approaches to address these pricing challenges. By leveraging historical data and advanced predictive modeling, it becomes possible to develop more sophisticated and accurate methods of smartphone price categorization and prediction.

I.2 Motivation

The motivation for this research stems from the increasingly complex and dynamic nature of the smartphone market. Several critical challenges drive the need for an advanced smartphone price categorization approach:

1. **Economic Significance for Manufacturers :** Inaccurate pricing can result in significant financial losses or missed market opportunities.
2. **Tranceparent Pricing for Consumers :** Many consumers face challenges in understanding the intrinsic value of smartphones. A data-driven approach to price categorization can provide transparent insights into the factors that genuinely influence smartphone pricing.

I.3 Objectives

The primary objectives of this project are:

1. **Develop a Robust Classification Model :**
   - Create a machine learning model capable of accurately categorizing smartphones into distinct price segments
   - Achieve high predictive accuracy using multiple classification algorithms
   - Identify and leverage the most significant features influencing smartphone pricing
2. **Feature Analysis and Selection :**
   - Conduct comprehensive analysis of smartphone features
   - Determine the most influential factors in price categorization
   - Develop a systematic approach to feature selection and importance ranking
3. **Comparative Algorithm Performance :**
   - Implement and evaluate multiple machine learning algorithms
   - Compare the performance of different classification techniques
   - Identify the most effective algorithm for smartphone price category prediction
4. **Practical Applicability :**
   - Demonstrate the practical utility of the developed model for both manufacturers and consumers

5. **Methodological Contribution :**
    - Develop a systematic approach to smartphone price categorization
    - Contribute to the existing body of knowledge in machine learning applications in market analysis
    - Establish a replicable methodology for similar predictive modeling challenges

SECTION II

LITERATURE REVIEW

In this paper Asim et al. [2] (2018) investigates price class prediction for smartphones using machine learning, focusing on determining whether a smartphone is economical or expensive. Data was collected from GSMArena, including features like display size, weight, RAM, and battery capacity. The dataset was preprocessed and categorized into four price classes: very economical, economical, expensive, and very expensive. Feature selection methods like InfoGain and WrapperAttributeEval were employed to reduce dataset dimensionality, optimizing computational efficiency. The study tested classifiers such as Decision Tree (J48) and Naïve Bayes, achieving a maximum accuracy of 78% with Decision Tree when combined with WrapperAttributeEval. Challenges included converting a regression problem into classification, which introduced errors, and the limited dataset size impacting accuracy. The paper concludes with recommendations for improved feature selection techniques, larger datasets, and extending the model for other product categories.

Chandrashekhara et al. [3] (2019), focuses on predicting smartphone prices using machine learning techniques like Support Vector Regression (SVR), Backpropagation Neural Network (BNN), and Multiple Linear Regression (MLR). The dataset included features such as brand, RAM, memory, battery power, and display size, with 262 records spanning 2010–2018, sourced from e-commerce platforms. Data preprocessing steps included handling missing values, standardizing formats, and splitting the data into training (80%) and testing (20%) sets. Performance metrics such as R-squared and correlation values were used to evaluate models. SVR performed the best with an R-squared of 0.86 and correlation of 0.93. Graphical analyses demonstrated that SVR had the most accurate predictions, followed by BNN and MLR. The authors highlight the potential of SVR for broader applications in price prediction across retail industries, suggesting

scalability through distributed systems like Hadoop.

Ercan & Şimşek et al. [4] (2023) examines the classification of smartphone prices into low, medium, high, and very high categories using a Kaggle dataset with 2000 entries and 20 features. Features included physical attributes (weight, dimensions), performance metrics (RAM, processor cores), and functionalities (3G, Wi-Fi). Four machine learning algorithms—Logistic Regression, Support Vector Machine (SVM), Decision Tree, and K-Nearest Neighbors (KNN)—were applied and evaluated. SVM achieved the highest accuracy (96.16%), outperforming Logistic Regression (91%), Decision Tree (82%), and KNN (41%). Confusion matrices and performance metrics (accuracy, precision, recall, and F1-score) were used to validate results. The study emphasizes the superiority of SVM for this classification problem and suggests exploring additional algorithms and datasets for improved accuracy in future work.

In another study, Abbasi et al. [5] (2024) focuses on predicting the price of second-hand electronic devices, particularly smartphones, using machine learning (ML) techniques. It addresses the growing trend of buying used electronics, driven by economic factors and technological advancements. The authors utilized web scraping to gather a dataset covering the last five years, containing various features influencing smartphone prices. They experimented with three ML algorithms—Random Forest, Linear Regression, and Multi-Layer Perceptron—to develop a predictive model. Performance evaluation was based on metrics like Absolute Percentage Difference (APD) and Root Mean Square Error (RMSE). Random Forest emerged as the best-performing model, demonstrating the lowest prediction error and superior generalization. The study highlights how accurate price prediction can aid both buyers and sellers, making the market more efficient. Data preprocessing and feature selection were critical steps, leading to a dataset with 24 variables. Techniques like 10-fold cross-validation were used to ensure robustness. The findings suggest that incorporating modern ML methods can significantly enhance price prediction accuracy compared to traditional approaches.

In this paper, Akash et al. [6] (2020) explores smartphone price prediction using machine learning, aiming to categorize smartphones as economical or expensive based on their features. The study emphasizes the relevance of predictive modeling in aiding marketing and consumer decision-making. Key features considered include processor type, battery capacity, memory, and screen size. The authors employed algorithms like Linear Regression and K-Nearest Neighbors (KNN) to build predictive models. The seven-step machine learning process involved data gathering, preparation, model selection, training, evaluation, hyperparameter tuning, and prediction. The KNN model showed higher accuracy compared to Linear Regression, particularly in scenarios involving complex, non-linear relationships. The research demonstrated the importance of feature selection to reduce computational complexity and enhance prediction precision. The conclusion underscores the impact of ML-driven price prediction in optimizing marketing strategies and enhancing

business decisions.

Finally, this paper Kumuda S et al. [7] (2021) investigates the prediction of smartphone prices using machine learning, with a focus on market analysis and consumer behavior. The study outlines the challenges of determining accurate smartphone prices due to diverse features and competitive market dynamics. A dataset containing information on screen size, memory, camera quality, and battery was used to train predictive models. Key algorithms employed included KNN, Forward Selection, and Backward Selection to manage data complexity and optimize feature selection. The data collection phase involved gathering specifications from various smartphones to construct a comprehensive dataset. The authors emphasized the role of data visualization techniques, like the Elbow Method, to identify the optimal number of features. The study presented a detailed analysis of feature importance, highlighting key parameters influencing price prediction. Classification and testing phases assessed the model's accuracy, using preprocessed datasets for training and evaluation. Results were visualized using graphs that compared predicted prices against specifications like RAM and memory. The conclusion suggests that precise feature selection can improve predictive accuracy and aid in product launch decisions.

SECTION III

DATASET DESCRIPTION

III.1 Source

The dataset for this smartphone price classification project was obtained from Kaggle, a prominent platform for data science and machine learning datasets.

Link - https://www.kaggle.com/datasets/iabhishekofficial/smartphone-price-classification

The dataset is publicly available and contains 2000 smartphone entries with 20 feature variables and a target variable representing price range.

III.2 Features

The dataset comprises 20 features that describe various characteristics of smartphones. Following table provides a detailed breakdown of these features:

Feature Name	Description	Type
battery_power	battery capacity in mAh	Numerical
blue	has bluetooth or not	Binary
clock_speed	speed at which processor executes instructions	Numerical
dual_sim	has dual sim support or not	Binary
fc	front Camera Megapixels	Numerical
pc	primary Camera Megapixels	Numerical
four_g	has 4G or not	Binary
three_g	has 3G or not	Binary
wifi	has wifi or not	Binary
int_memory	internal Memory capacity	Numerical
m_dep	smartphone Depth in cm	Numerical
mobile_wt	weight of the smartphone	Numerical
n_cores	number of cores in processor	Numerical
px_height	pixel Resolution Height	Numerical
px_width	pixel Resolution Width	Numerical
ram	RAM in MB	Numerical
touch_screen	has touch screen or not	Binary
sc_h	screen Height in cm	Numerical
sc_w	screen Width in cm	Numerical
talk_time	longest time that a single battery charge will last over a call	Numerical

Among these features, some are binary variables (e.g., blue, dual_sim, four_g, three_g touch_screen, wifi), while others are continuous variables (e.g., battery_power, clock_speed, int_memory, ram, talk_time). These features collectively provide a comprehensive overview of the smartphone's technical specifications and capabilities.

A histogram visualization of all individual features is given below:

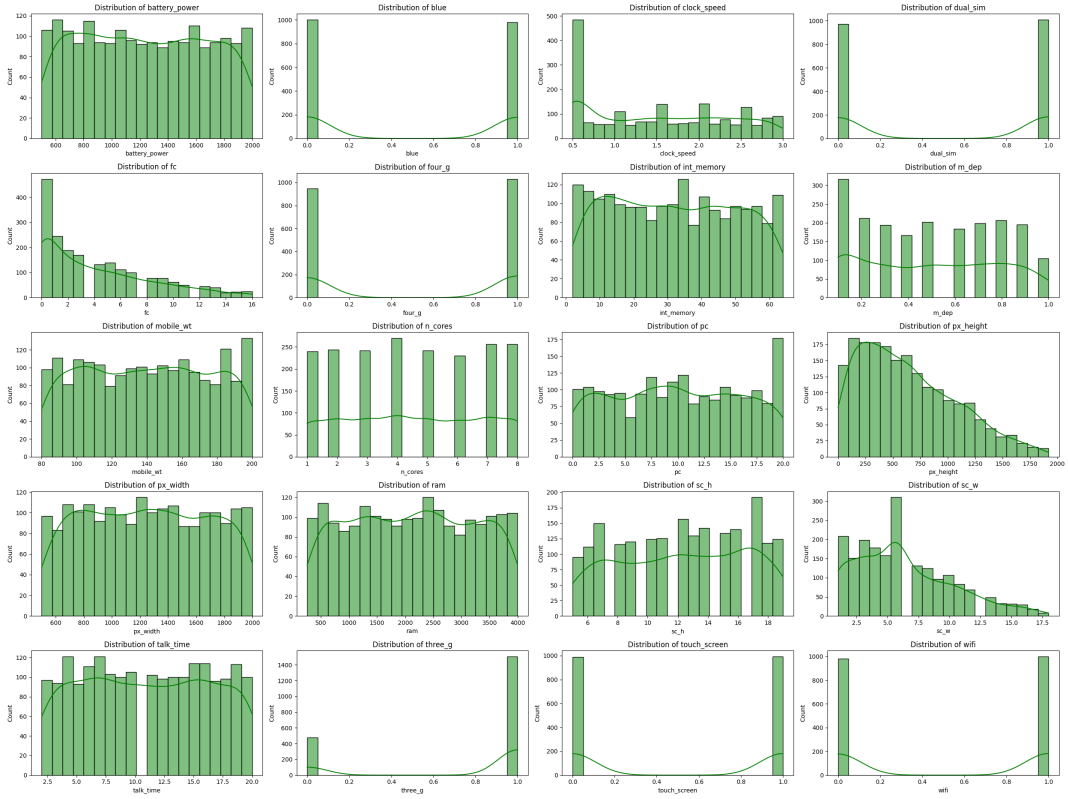

Figure III.1: Histogram visualization of all features.

This visualization give us insight into the distributions of the features. An important observation here is that px_height can be normalized. Most of the categorical features are balanced except three_g.

III.3 Target Variable

The target variable, price_range, represents the categorization of smartphones into different price segments. It is a categorical variable with four distinct classes:

- `0` : Low-cost smartphones
- `1` : Medium-low cost smartphones
- `2` : Medium-high cost smartphones
- `3` : High-end, premium smartphones

SECTION IV

DATA PREPROCESSING

IV.1 Data Cleaning

IV.1.1 Handling Missing Values

A check for missing values - `df.isnull().sum()` revealed no missing values in the dataset.

IV.1.2 Handling Duplicate Values

Also, the dataset was checked for duplicate entries - `df.duplicated().sum()`. No duplicate entries were found.

IV.1.3 Handling Invalid Values

The dataset was checked for negative entries and none were found in the dataset. There are some features which can not be zero, like battery_power, ram, etc. So, we checked for zero values in these features. We saw px_height and sc_w are have 2 and 180 zero values respectively. We will replace these zero values with the mean of the respective features.

Note that fc and pc are numerical yet zero values are not invalid. They can be zero if the phone does not have a front or primary camera.

IV.2 Outlier Handling

Outliers can significantly impact the performance of machine learning models. To identify and handle outliers, box plots were generated for each feature to visualize the distribution of data points.

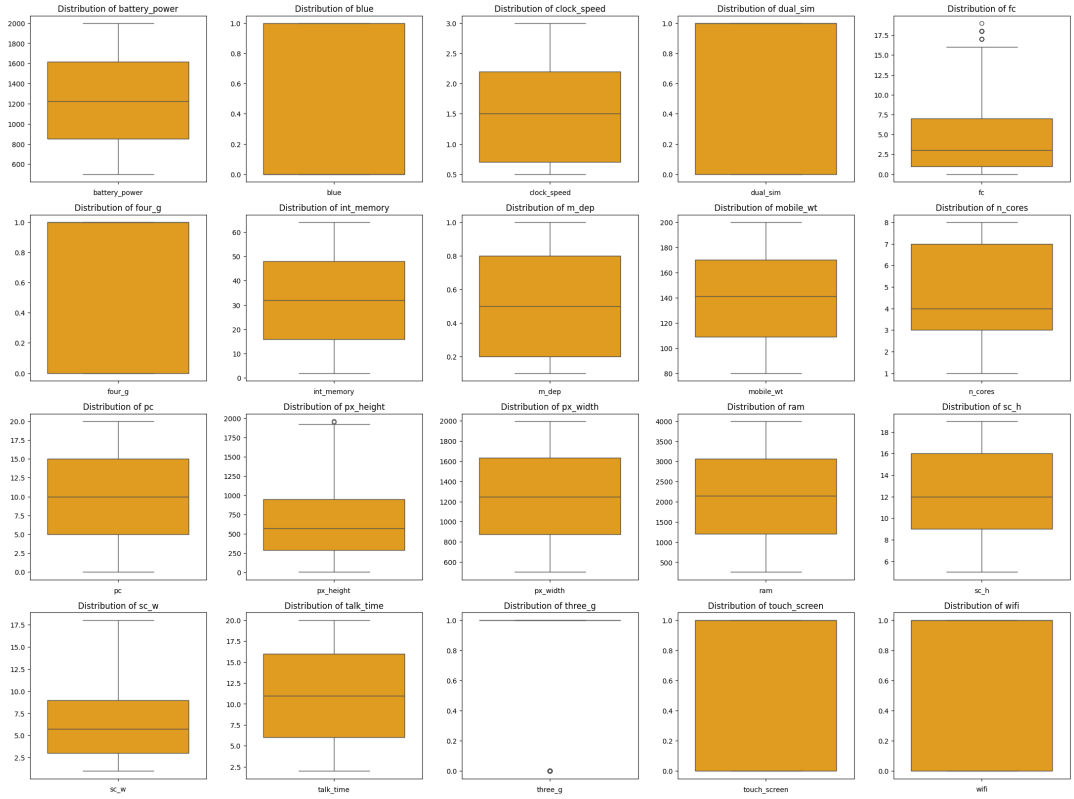

Figure IV.1: Box plots of features before outlier handling.

The box plots revealed outliers in two features, fc and px_height. To address these outliers, the IQR (Interquartile Range) method was used to detect and remove extreme values.

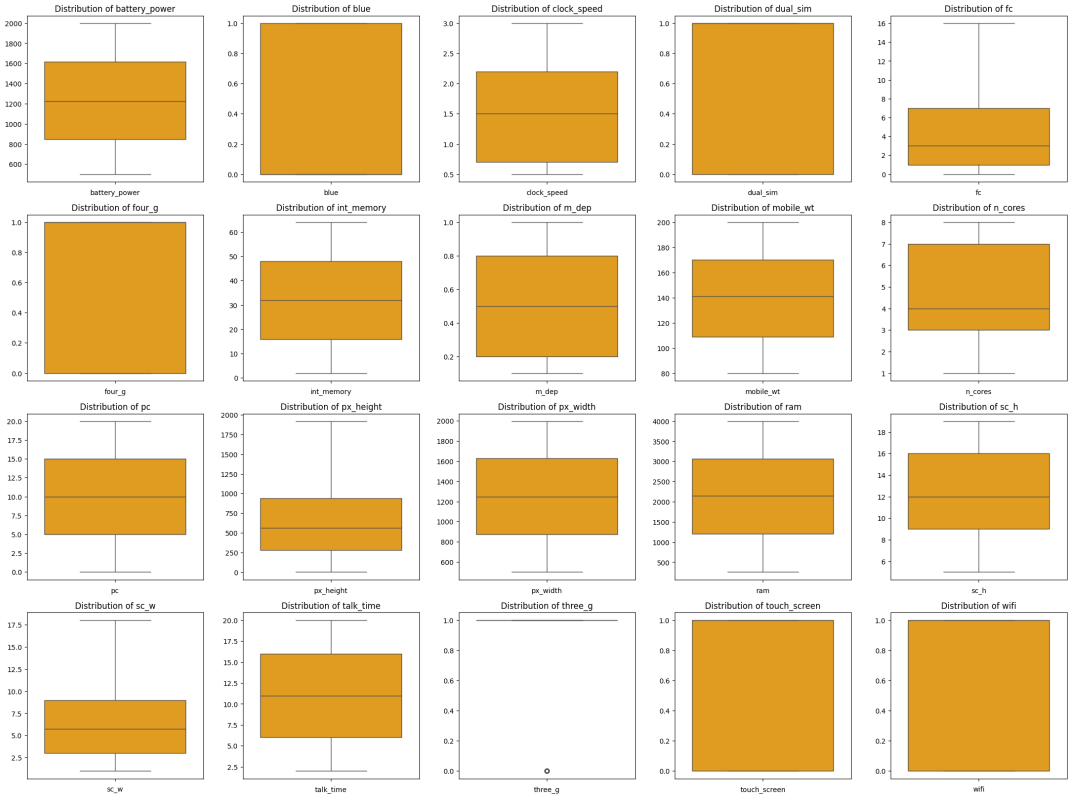

Figure IV.2: Box plots of features after outlier handling.

IV.3 Class Imbalance Check

Class imbalance can affect the performance of classification models. To check for class imbalance, the distribution of the target variable was visualized using a count plot.

We saw that the dataset was balanced across all four price range categories.

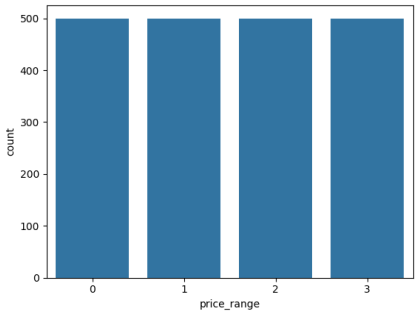

Figure IV.3: Class imbalance check for target variable.

The count plot revealed that the dataset was balanced across all four price range categories, with an equal distribution of samples in each class.

IV.4 Correlation Analysis

A correlation matrix was generated to identify the relationships between different features and the target variable. This analysis helped identify the most influential features in predicting smartphone prices. The correlation matrix was visualized using a heatmap to provide a clear overview of feature relationships.

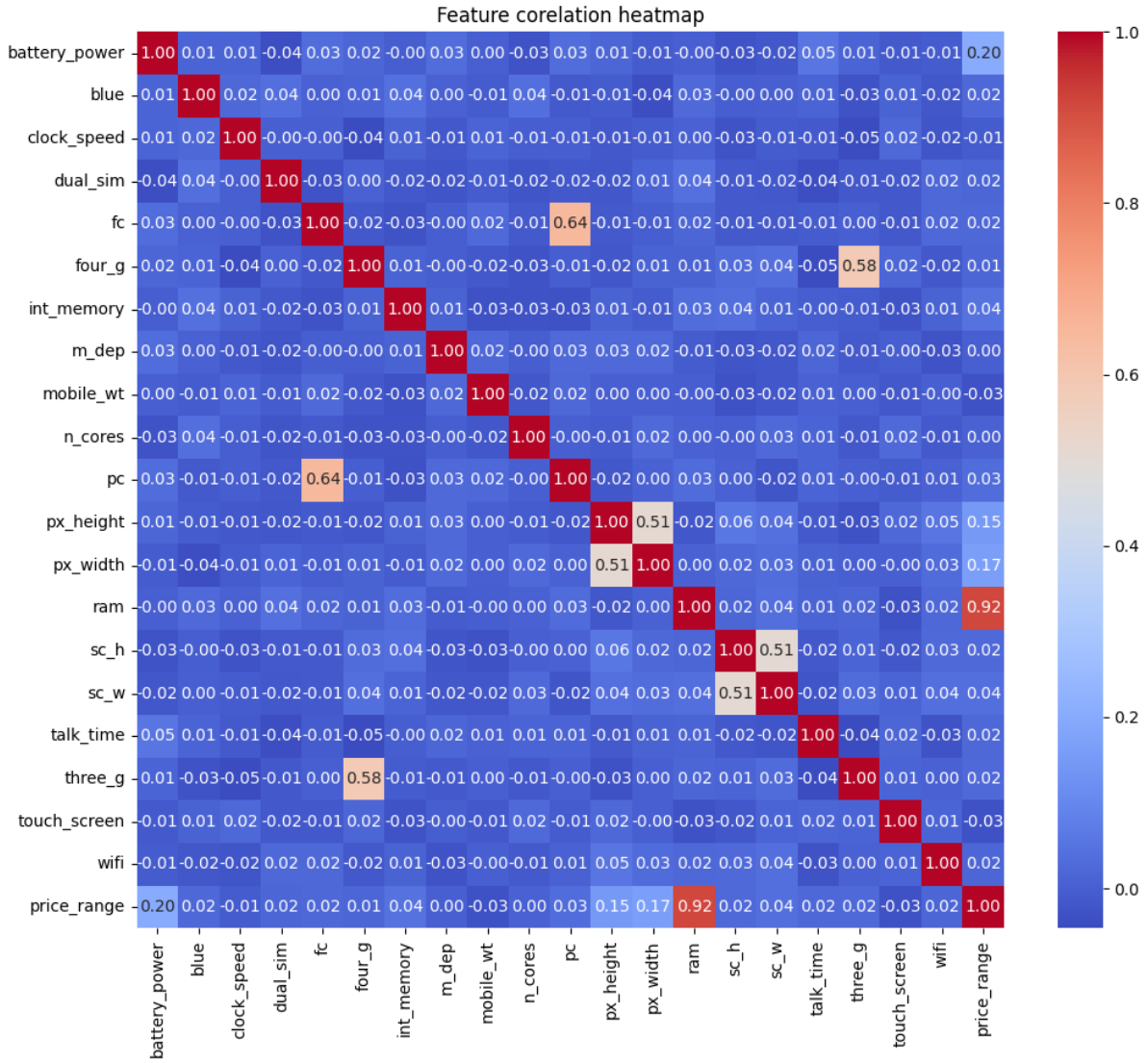

Figure IV.4: Correlation heatmap of features.

Here we saw that ram is highly correlated with price range. There are some features which are correlated with each

other like:

1. **3G and 4G :** A high correlation here suggests that devices with 4G almost always support 3G, making one of these features redundant.
2. **fc and pc :** These features are correlated, as better primary cameras often accompany better front cameras.
3. **px_height and px_width :** These are components of screen resolution and are naturally correlated.
4. **sc_h and sc_w :** These are also naturally correlated.

These correlations will be taken into note during next steps to reduce feature redundancy and improve model performance.

IV.5 Feature Engineering

Feature engineering involved creating new features from existing ones to enhance the dataset's predictive power. The goal was to derive meaningful attributes that capture relationships between variables and optimize the dataset for machine learning model training. Two new features were introduced based on the existing data:

1. **px_area :** The total pixel area of the smartphone screen, calculated as the product of pixel resolution height (px_height) and pixel resolution width (px_width).
2. **screen_area :** The physical screen area of the smartphone, calculated as the product of screen height (sc_h) and screen width (sc_w).

After deriving the new features, the original components (px_height, px_width, sc_h, sc_w) were removed from the dataset: Next, based on correlation analysis, the following redundant features were removed:

- A high correlation exists between 3G and 4G, as devices with 4G generally support 3G. The 4G feature was retained, and 3G was removed.
- Front and primary cameras are correlated, as primary cameras often accompany front cameras. The pc feature was retained, and fc was removed.

The correlation matrix after feature engineering is as follows:

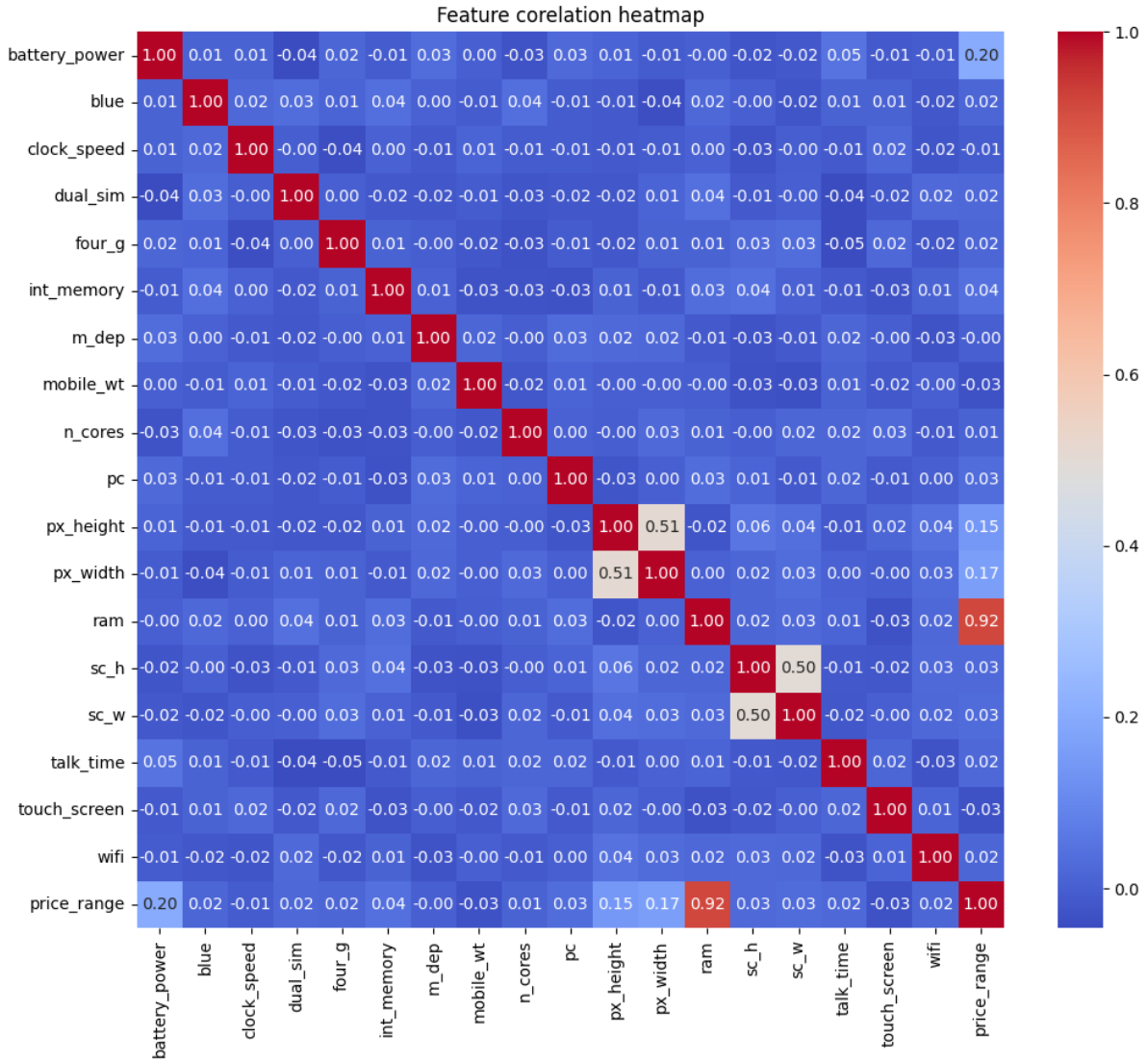

Figure IV.5: Correlation heatmap of features.

IV.6 Feature Selection

We preformed feature selection to identify the most relevant features for predicting smartphone prices. This step involved analyzing feature importance scores from machine learning models, conducting correlation analysis, and using domain knowledge to select the most influential features. Feature selection is aimed to reduce model complexity and improve prediction accuracy.

In our project, we used the ANOVA F-test (Analysis of Variance) method to evaluate each feature's relationship with the target variable, price_range, and select features that are statistically significant. The threshold for selection is a p-value of

less than 0.1, indicating a 90% confidence level.

The `SelectKBest` method from `sklearn.feature_selection` is used with the scoring function `f_classif`, which applies the ANOVA F-test.

Initially, `k='all'` is set to retain all features for evaluation, allowing for the computation of statistical metrics across all features.

After fitting the model, the p-values and F-scores for each feature are extracted. These metrics help assess the significance and contribution of each feature to the target variable.

Features with a p-value less than 0.05 are considered statistically significant and are selected for further analysis.

The selected features are:

`['battery_power', 'int_memory', 'mobile_wt', 'n_cores', 'ram', 'px_area', 'screen_area']`

These are then used for model training and evaluation to determine their impact on prediction accuracy and model performance.

IV.7 Normalizing features

A historam plot of the features after feature selection was done as follows:

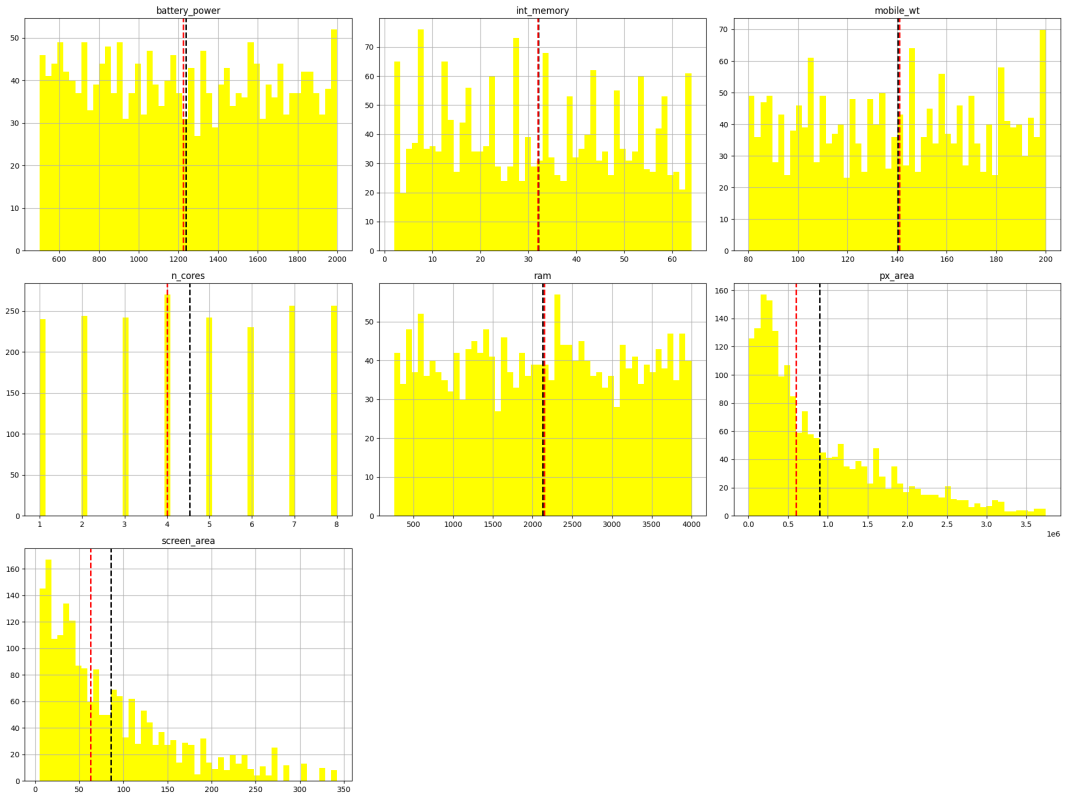

Figure IV.6: Histogram visualization of all features after feature selection

As we can see the px_area and screen area can be normalized.
We will normalize these features using `PowerTransformer()`.

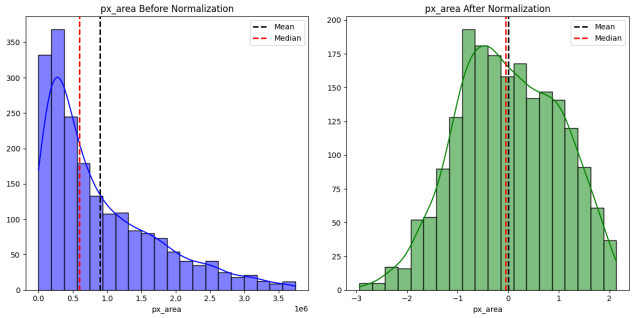

Figure IV.7: Histogram visualization of px_area before and after normalization

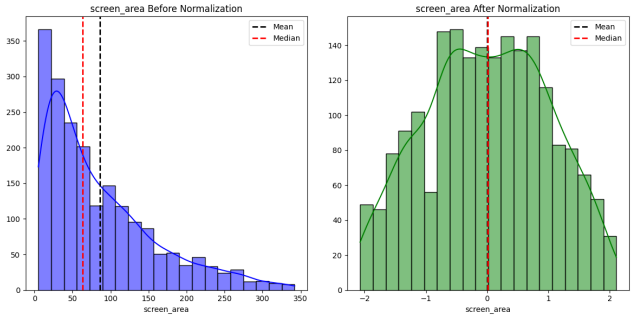

Figure IV.8: Histogram visualization of screen_area before and after normalization

IV.8 Train-Test Split

The dataset was split into training and testing sets to evaluate model performance. The training set was used to train the machine learning models, while the testing set was used to assess model accuracy and generalization. The split ratio was 80% training and 20% testing to ensure an adequate balance between model training and evaluation. We used sklearn's `train_test_split` function to perform the split.

IV.9 Scaling

The dataset was scaled using the `StandardScaler` from `sklearn.preprocessing` to normalize the feature values. Scaling ensures that all features contribute equally to the model training process and prevents any feature from dominating the others. The scaled dataset was used for model training and evaluation to improve prediction accuracy and model performance.

SECTION V

METHODOLOGY

V.1 Algorithms Used

For this project, we experimented with several machine learning algorithms to predict smartphone price categories. The following algorithms were implemented and evaluated:

1. **Logistic Regression:** Logistic Regression allows us to predict the probability of the dependent variable from a given set of independent variables. The probability values are between 0 and 1. [1]

2. **K-Nearest Neighbors:** K-Nearest Neighbors is a well-known classification technique that bases predictions on finding the closest neighbors in classes that share a lot of characteristics. Since the dataset is scanned one by one to find the nearest neighbors, the performance of the algorithm decreases. It is also known as the lazy learning method. It works slowly in large volumes of data. [1]

3. **Random Forest:** Random Forest is a machine learning algorithm that creates a large number of decision trees and combines their predictions. It is a type of ensemble learning method that can be used for classification and regression. Random Forest is a versatile algorithm that can be used for a variety of tasks, including classification, regression, and feature selection. [1]

4. **Decision Tree:** There are decision and leaf nodes according to the goal and independent variables in the decision tree algorithm. Because it is a categorization technique that produces a tree-like structure, it is known as a decision tree. The data set entries are processed into a tree to be used for classification, after which the classification procedure is carried out. Algorithms can follow different paths in the selection of root, node and branching criteria.

[1]

5. **Support Vector Machine:** In Support Vector Machine, the data is separated into two classes and placed on a plane known as the marginal plane. Points from two classes that are adjacent to the line are called support vectors. The goal of Support Vector Machine is to choose the marginal plane with the greatest separation between two data points. This classifier is called a linear support vector machine if the training data can be separated linearly, albeit coarsely. If the data are not linearly separated, it is preferable to use kernel methods and soft range maximization to obtain nonlinear support vector machines. [1]
6. **XGBoost:** XGBoost is a scalable and accurate implementation of gradient boosting machines. It is an efficient and effective algorithm that can be used for regression, classification, ranking, and user-defined prediction problems. XGBoost is known for its speed and performance, making it a popular choice for machine learning tasks. [1]

V.2 Justification

The selection of these algorithms was based on their suitability for classification tasks, performance in handling multi-class classification problems, and ability to capture complex relationships in the data. The justification behind using each of these algorithms are as follows:

1. **Logistic Regression:** Logistic Regression serves as a great starting point for classification tasks. For predicting price ranges, it can handle categorical output well and provides insights into which features (e.g., RAM or battery power) contribute most to the classification. This helps establish a baseline performance for comparison with other models.
2. **K-Nearest Neighbors:** KNN is effective in scenarios where decision boundaries are non-linear, as is often the case with diverse smartphone features like camera quality and battery power. For this project, it helps to explore the relationship between clusters of data points and their price ranges.
3. **Random Forest:** Random Forest excels in handling mixed data types (e.g., numerical and categorical) and mitigating overfitting. In this project, its feature importance analysis can identify critical smartphone features influencing price ranges, such as internal storage or RAM.
4. **Decision Tree:** Decision Trees provide an interpretable model for understanding hierarchical decision-making based on features. For this project, it illustrates how smartphone features interact to influence price categories, aiding feature selection and refinement.
5. **Support Vector Machine:** SVM can capture complex relationships between features using kernel tricks, making it suitable for handling overlapping data distributions. Given the challenge of imbalanced class distributions, SVM with proper class weighting can improve prediction accuracy.

6. **XGBoost:** XGBoost's advanced boosting mechanism is ideal for capturing intricate patterns in the data. For this project, it addresses imbalanced classes effectively through weight adjustment, delivering state-of-the-art performance on complex datasets.

SECTION VI

IMPLEMENTATION

VI.1 Tools and Libraries

The tools and libraries used for this project are:

1. **Python:** The primary programming language for this project.
2. **Jupyter Notebook:** For interactive code development.
3. **Data Handling Libraries:**
   (a) **Pandas:** For data manipulation, cleaning, and handling structured datasets.
   (b) **Numpy:** For numerical operations, array manipulations, and faster mathematical computations.
4. **Visualization Libraries:**
   (a) **Matplotlib:** For creating static visualizations like bar charts and line graphs etc.
   (b) **Seaborn:** For advanced visualizations, particularly heatmaps and confusion matrices.
5. **Machine Learning Libraries:**
   (a) **Scikit-learn:** For building, training, and evaluating machine learning models (Logistic Regression, KNN, Random Forest, Decision Tree, Voting Classifier, and SVM). It has also been used for data preprocessing, data splitting, and feature selection.
   (b) **XGBoost:** For building and training gradient boosting models.
6. **Other Libraries:**

(a) **TQDM:** For showing progress bars during loops.

(b) **Time:** For tracking the time taken by different parts of the code.

VI.2 Parameters

At first, the models were initialized with default parameters from their respective classes in Scikit-learn and XGBoost. Notably:

1. **Logistic Regression:** `max_iter=1000` was set to ensure convergence during optimization.
2. **SVM:** `probability=True` was used to enable probabilistic predictions for compatibility with ensemble models.
3. **XGBoost:** Used `eval_metric='mlogloss'` as the evaluation metric.

The initial configurations resulted in good training accuracy but some inconsistencies in test performance. Hence hyperparameter tuning was performed to optimize the models for better generalization and prediction accuracy.

The best configuration we got from hyperparameter tuning is as follows:

Model	**Best Parameters**	**Best Score**
Logistic Regression	{'classifier__C': 10, 'classifier__max_iter': ...}	0.9369
KNN	{'classifier__n_neighbors': 7, 'classifier__p': ...}	0.7740
Random Forest	{'classifier__max_depth': None, 'classifier__m': ...}	0.8946
Decision Tree	{'classifier__criterion': 'gini', 'classifier__': ...}	0.8567
SVM	{'classifier__C': 10, 'classifier__gamma': 'sc': ...}	0.9350
XGBoost	{'classifier__learning_rate': 0.1, 'classifier__': ...}	0.8977

Table VI.1: Best Parameters and Scores for each Model

The best score represents the highest cross-validation score achieved by each model during hyperparameter tuning. This score is typically the mean accuracy (or another evaluation metric) obtained from cross-validation, indicating how well the model is expected to perform on unseen data with the best-found hyperparameters.

VI.3 Training Process

As discussed in Section IV.8, the dataset was split into training and testing sets.
The training set was used to train various machine learning models, while the testing set was used to evaluate their performance. The models included Logistic Regression, K-Nearest Neighbors (KNN), Random Forest, Decision Tree, Support Vector Machine (SVM), and XGBoost.
Each model was trained using the scaled training data and then evaluated on the scaled test data to assess their accuracy, precision, recall, and F1-score. The training process involved fitting the models to the training data and making predictions on both the training and test data. The predictions were compared with the actual labels to calculate the evaluation metrics. This process was repeated for each model to compare their performance and identify the best model for predicting smartphone prices.
After this the models were hyperparameter tuned using `GridSearchCV` to get the best parameters for each model. The models were then retrained using the best parameters and evaluated again to get the final performance metrics.

The results were stored in a DataFrame for better visualization and analysis.

SECTION VII

RESULTS

VII.1 Plots

In this section, we present the confusion matrices, learning curves and ROC-AUC curves for the various machine learning models used in this study.

Learning curves illustrate the model's performance on the training and validation sets as the training size increases. This helps us understand how well the model generalizes to unseen data and whether it is overfitting or underfitting.

Confusion matrices provide a detailed breakdown of the model's performance by showing the number of true positive, true negative, false positive, and false negative predictions. This allows us to gain deeper insights into the strengths and weaknesses of each model, particularly in terms of their ability to correctly classify instances of each class.

ROC-AUC curves provide a visual representation of the model's true positive rate against the false positive rate, helping us evaluate the model's performance across different thresholds.

Precision-Recall curves illustrate the trade-off between precision and recall for different threshold values, providing insights into the model's ability to balance false positives and false negatives.

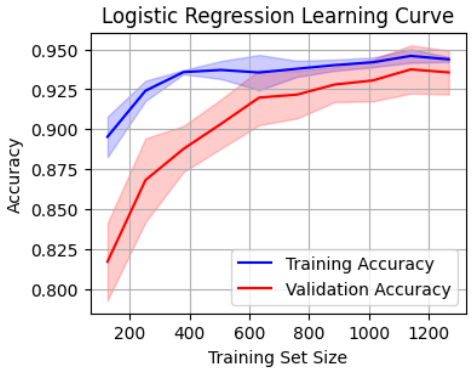

Figure VII.1: Learning Curve for Logistic Regression before Hyperparameter Tuning

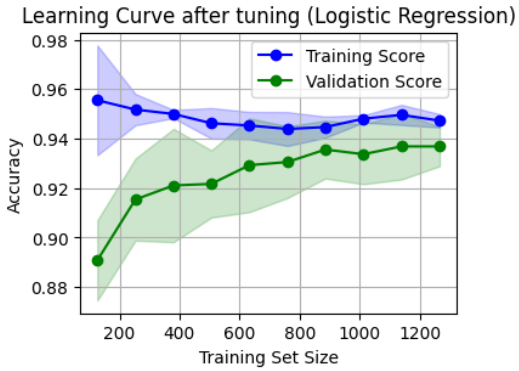

Figure VII.2: Learning Curve for Logistic Regression after Hyperparameter Tuning

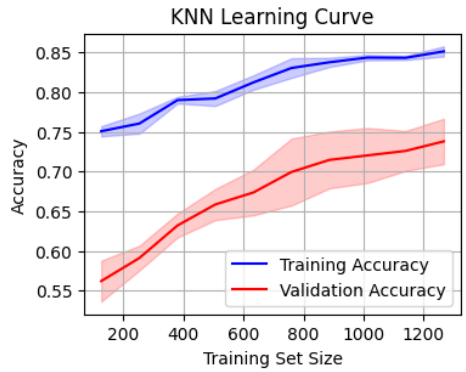

Figure VII.3: Learning Curve for K-Nearest Neighbors (KNN) before Hyperparameter Tuning

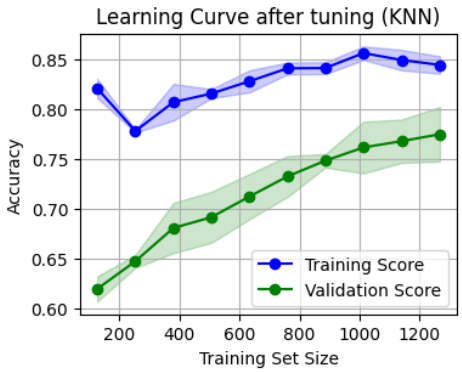

Figure VII.4: Learning Curve for K-Nearest Neighbors (KNN) after Hyperparameter Tuning

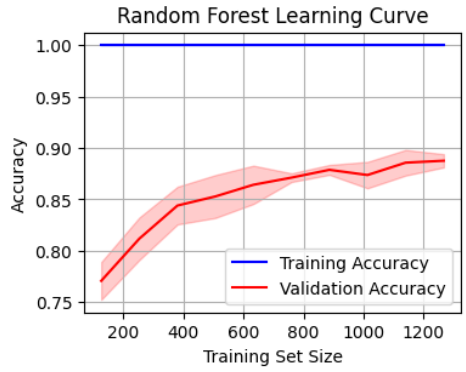

Figure VII.5: Learning Curve for Random Forest before Hyperparameter Tuning

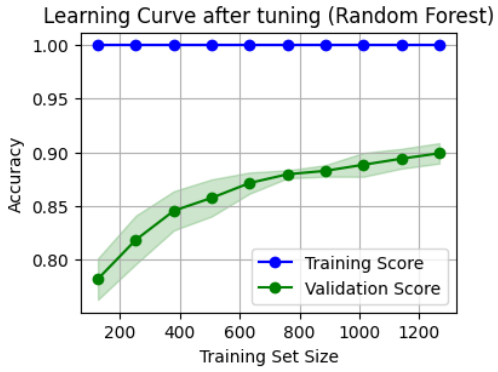

Figure VII.6: Learning Curve for Random Forest after Hyperparameter Tuning

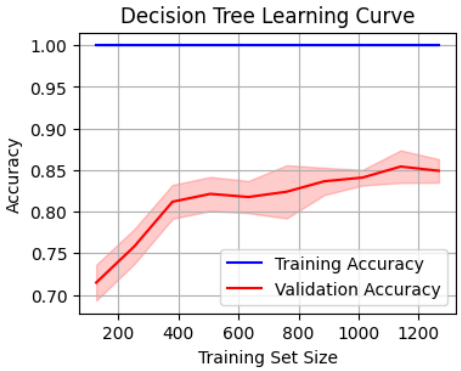

Figure VII.7: Learning Curve for Decision Tree before Hyperparameter Tuning

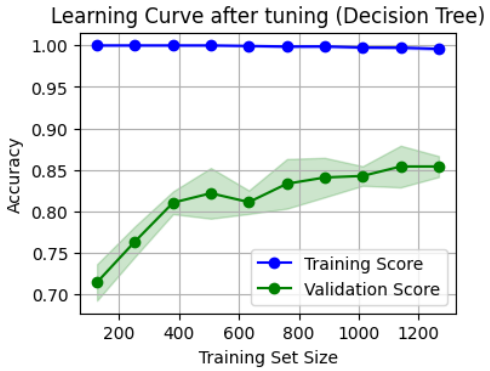

Figure VII.8: Learning Curve for Decision Tree after Hyperparameter Tuning

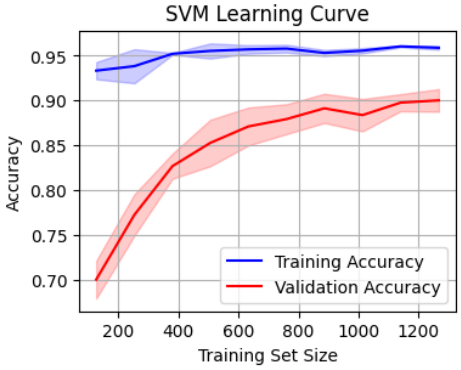

Figure VII.9: Learning Curve for Support Vector Machine (SVM) before Hyperparameter Tuning

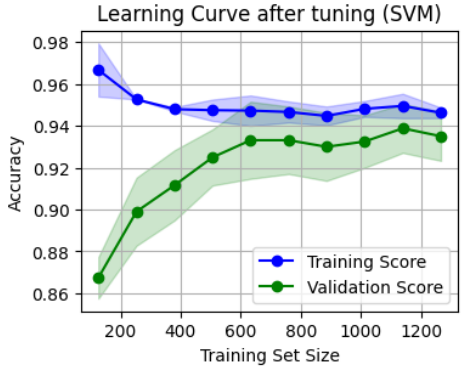

Figure VII.10: Learning Curve for Support Vector Machine (SVM) after Hyperparameter Tuning

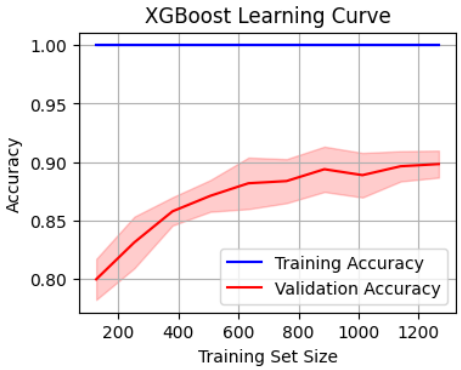

Figure VII.11: Learning Curve for XGBoost before Hyperparameter Tuning

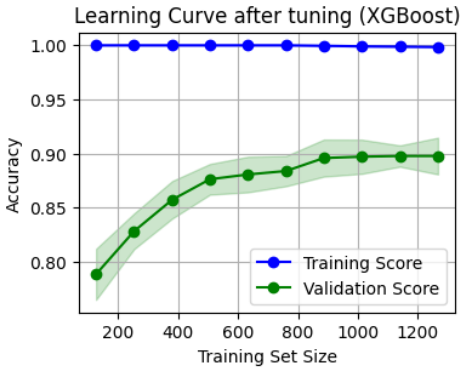

Figure VII.12: Learning Curve for XGBoost after Hyperparameter Tuning

By analsing these learning curves we infer the following:

- **Logistic Regression**:
  - **Before Tuning**: Moderate gap between training and validation accuracy, with validation scores plateauing slightly lower.
  - **After Tuning**: Reduced gap, with validation accuracy closely tracking training accuracy, suggesting a well-generalized model.
- **KNN**:
  - **Before Tuning**: Significant overfitting is evident as training accuracy is very high while validation accuracy lags.
  - **After Tuning**: Validation accuracy improves significantly, and the gap narrows, suggesting that the optimal number of neighbors was found.
- **Random Forest**:
  - **Before Tuning**: Overfitting is apparent; training accuracy is high, but validation accuracy is much lower and plateaus early.
  - **After Tuning**: Validation accuracy improves substantially, with a narrower gap between training and validation curves, showing better generalization.
- **Decision Tree**:
  - **Before Tuning**: Severe overfitting with perfect training accuracy but low validation accuracy.
  - **After Tuning**: Validation accuracy improves considerably, with a more realistic and generalizable training score.
- **SVM**:
  - **Before Tuning**: Moderate gap between training and validation accuracy; validation scores plateau at a sub-optimal level.
  - **After Tuning**: Narrowed gap and improved validation accuracy indicate better parameter selection.
- **XGBoost**:
  - **Before Tuning**: Overfitting is evident as training accuracy is high, but validation accuracy is lower.
  - **After Tuning**: Dramatic improvement in validation accuracy and narrowing of the gap indicate effective hyperparameter tuning.

Next we will look at the confusion matrices.

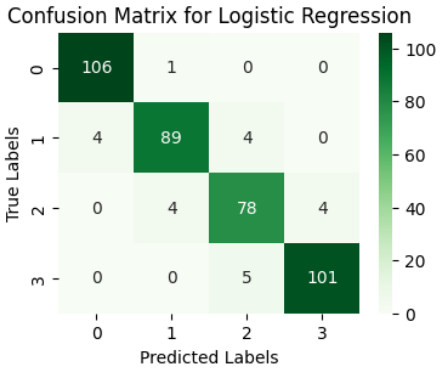

Figure VII.13: Confusion Matrix for Logistic Regression before Hyperparameter Tuning

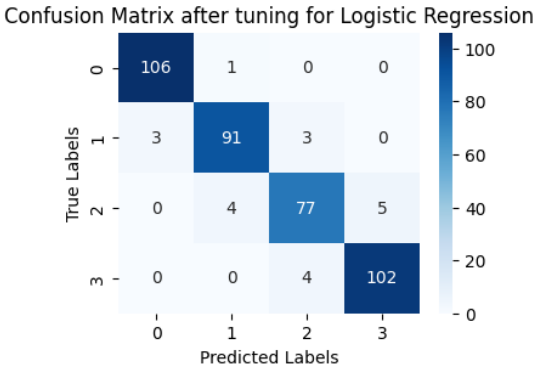

Figure VII.14: Confusion Matrix for Logistic Regression after Hyperparameter Tuning

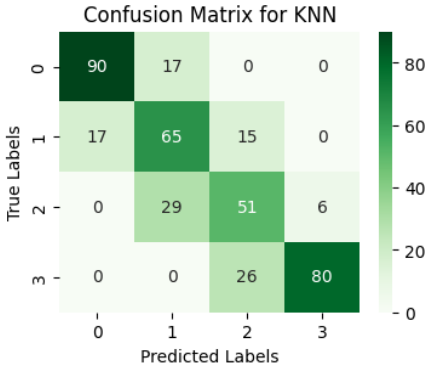

Figure VII.15: Confusion Matrix for K-Nearest Neighbors (KNN) before Hyperparameter Tuning

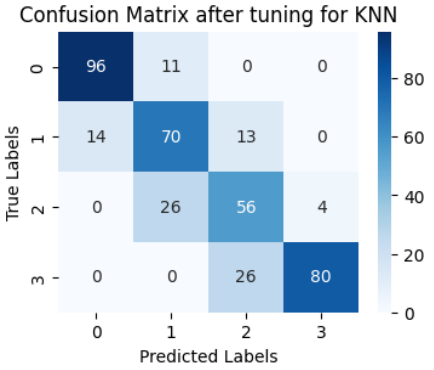

Figure VII.16: Confusion Matrix for K-Nearest Neighbors (KNN) after Hyperparameter Tuning

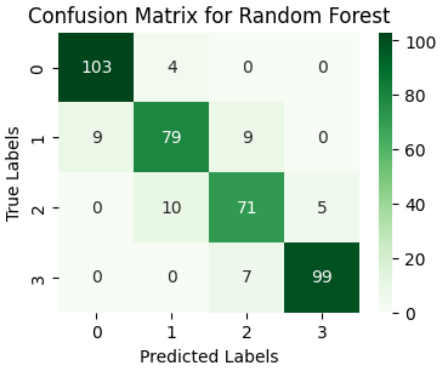

Figure VII.17: Confusion Matrix for Random Forest before Hyperparameter Tuning

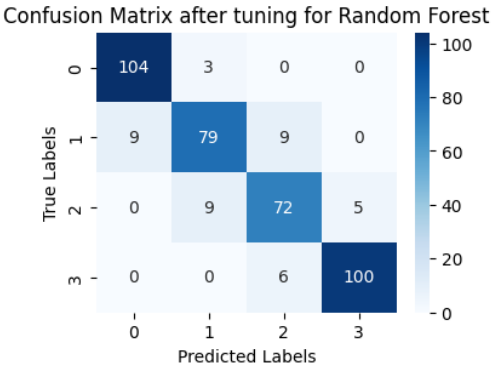

Figure VII.18: Confusion Matrix for Random Forest after Hyperparameter Tuning

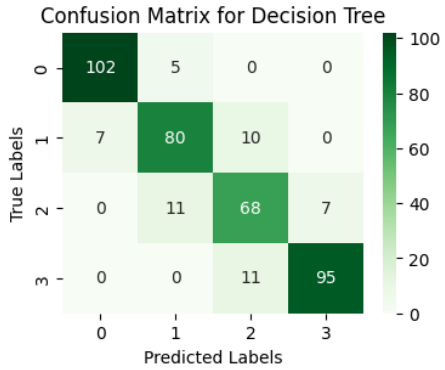

Figure VII.19: Confusion Matrix for Decision Tree before Hyperparameter Tuning

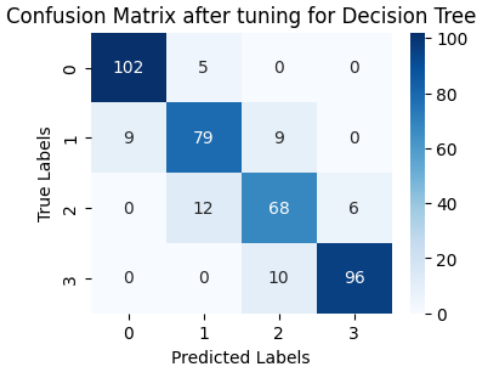

Figure VII.20: Confusion Matrix for Decision Tree after Hyperparameter Tuning

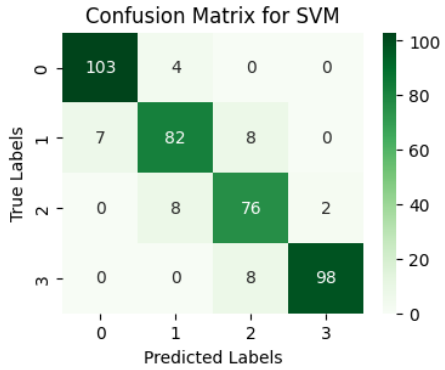

Figure VII.21: Confusion Matrix for Support Vector Machine (SVM) before Hyperparameter Tuning

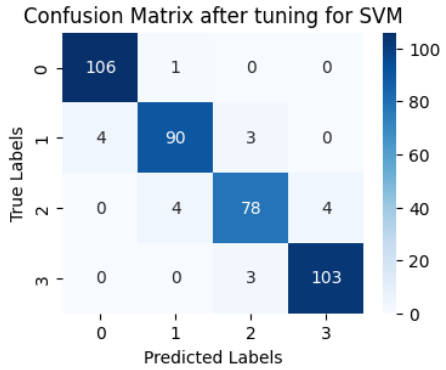

Figure VII.22: Confusion Matrix for Support Vector Machine (SVM) after Hyperparameter Tuning

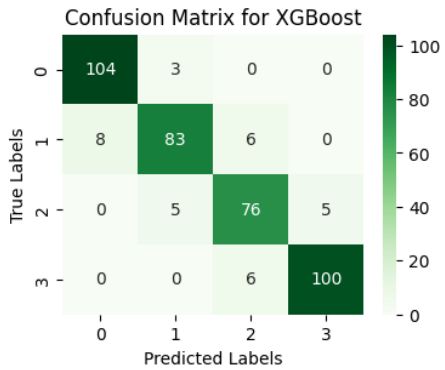

Figure VII.23: Confusion Matrix for XGBoost before Hyperparameter Tuning

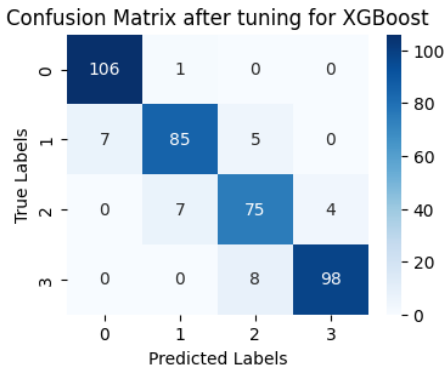

Figure VII.24: Confusion Matrix for XGBoost after Hyperparameter Tuning

By analyzing these confusion matrices, we can infer the following:

- **Logistic Regression**:
  - **Before Tuning**: Consistent performance with high true positives and low false positives.
  - **After Tuning**: Slight improvement in true positives for Class 1 and Class 3, indicating a well-optimized model.
- **KNN**:
  - **Before Tuning**: Moderate performance with higher false positives, especially for Class 1 and Class 2.
  - **After Tuning**: Significant improvement in reducing false positives for Class 0 and Class 1, indicating effective hyperparameter tuning.
- **Random Forest**:
  - **Before Tuning**: High true positives but some false positives, particularly for Class 1 and Class 2.
  - **After Tuning**: Slight changes, indicating that the model was already performing well.
- **Decision Tree**:
  - **Before Tuning**: High true positives but some false positives, particularly for Class 1 and Class 2.
  - **After Tuning**: Slight changes, indicating that the model was already performing well.
- **SVM**:
  - **Before Tuning**: Consistent performance with high true positives and low false positives.
  - **After Tuning**: Slight improvement in true positives for Class 1 and Class 3, indicating a well-optimized model.
- **XGBoost**:
  - **Before Tuning**: High true positives but some false positives, particularly for Class 1 and Class 2.
  - **After Tuning**: Slight changes, indicating that the model was already performing well.

Next we will look at the ROC-AUC and Precision-Recall curves.

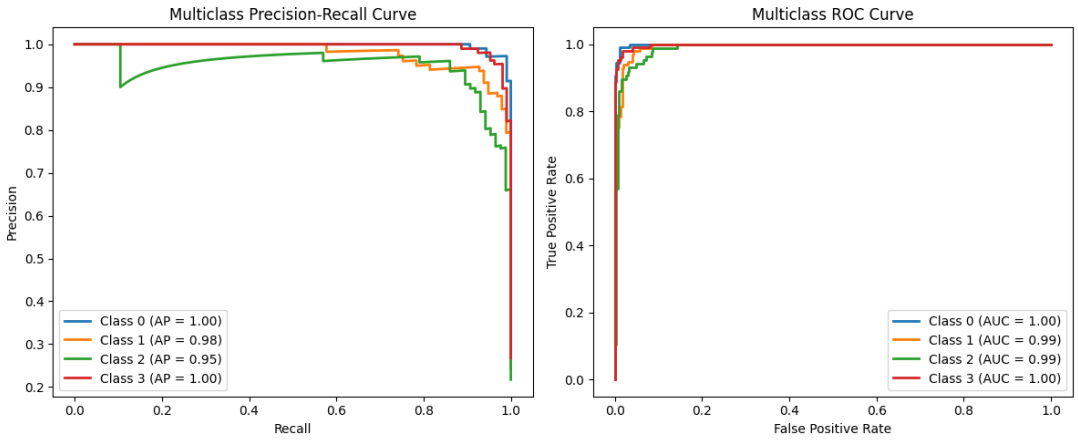

Figure VII.25: Precision-Recall and ROC-AUC curves for Logistic Regression

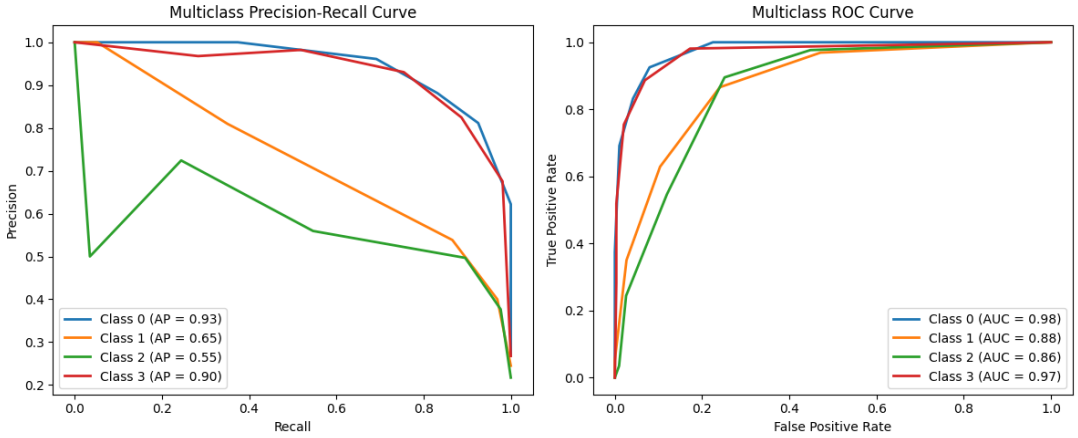

Figure VII.26: Precision-Recall and ROC-AUC curves for K-Nearest Neighbors (KNN)

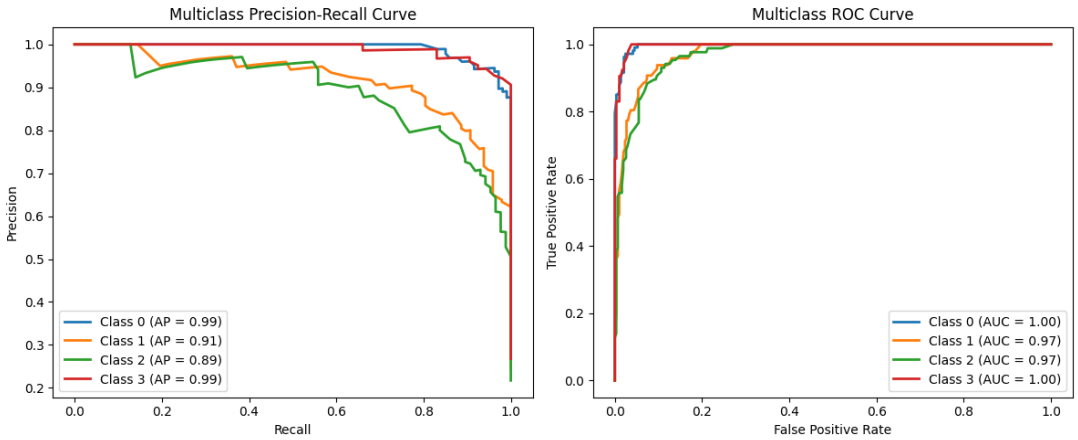

Figure VII.27: Precision-Recall and ROC-AUC curves for Random Forest

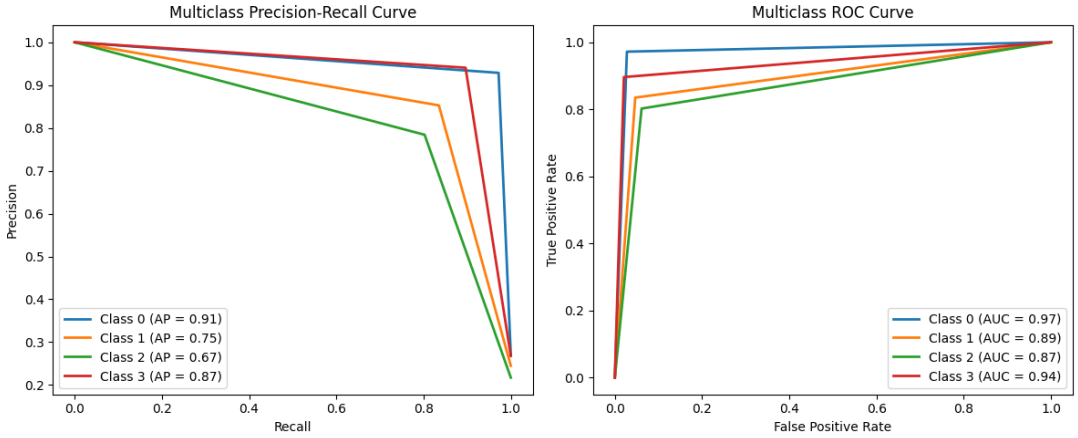

Figure VII.28: Precision-Recall and ROC-AUC curves for Decision Tree

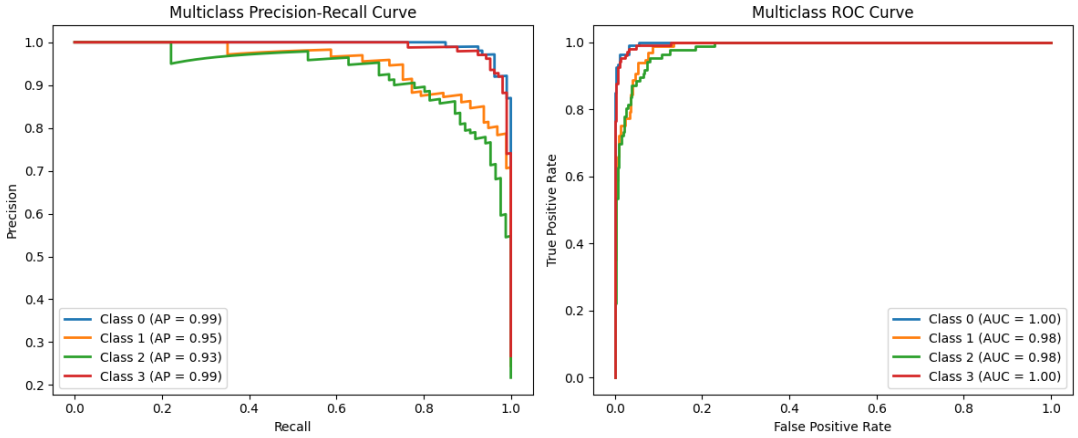

Figure VII.29: Precision-Recall and ROC-AUC curves for Support Vector Machine (SVM)

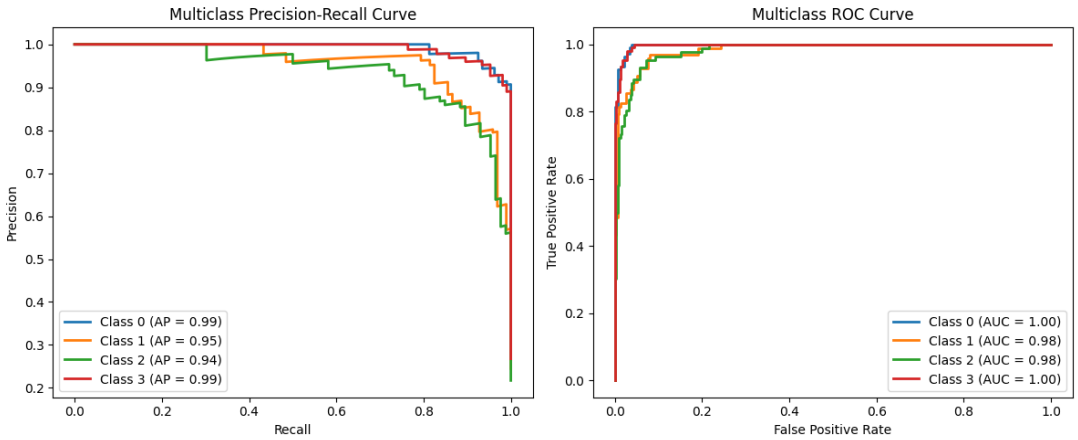

Figure VII.30: Precision-Recall and ROC-AUC curves for XGBoost

By analsing the above curves we infer the following:

- **Logistic Regression**:
  - **ROC-AUC**: The curves are tightly grouped, indicating good separation between the classes. The AUC values are relatively high for most classes, showing reliable performance in distinguishing between them.
  - **Precision-Recall**: The curve is above 0.5, which indicates that the model is effective, particularly for Class 0 (with the highest AUC).
- **KNN**:
  - **ROC-AUC**: The curves for some classes are lower compared to others, indicating less effective separation for certain classes. AUC values for some classes are below 0.9.
  - **Precision-Recall**: There is a noticeable drop for some classes, indicating that KNN struggles with precision at lower recall levels.
- **Random Forest**:
  - **ROC-AUC**: Shows strong performance with AUC close to 1.0 for most classes, suggesting that the model performs very well in distinguishing between the classes.
  - **Precision-Recall**: High curves for most classes, indicating strong precision, especially for Class 0.
- **Decision Tree**:
  - **ROC-AUC**: Shows some weaknesses with lower AUC values, particularly for Class 2 and Class 3, indicating less reliable separation.
  - **Precision-Recall**: The curves drop significantly for some classes, suggesting that the Decision Tree may not be as reliable in predicting positive cases for certain classes.
- **SVM**:
  - **ROC-AUC**: High AUC values for most classes, suggesting strong class separation. The performance is consistent across classes.
  - **Precision-Recall**: Shows good performance for most classes, indicating a high level of precision across various thresholds.
- **XGBoost**:
  - **ROC-AUC**: Excellent performance with AUC close to 1.0 for most classes, suggesting that XGBoost is particularly strong at distinguishing between classes.
  - **Precision-Recall**: Similar to Random Forest, XGBoost shows high precision-recall curves, indicating reliable prediction, particularly for Class 0 and Class 1.

VII.2 Evaluation Metrics

The initial model training resulted good accuracy during training but some inconsistencies in test performance, meaning there was overfitting.

Model	Accuracy Train	Accuracy Test	Precision Train	Precision Test	Recall Train	Recall Test	F1-Score Train	F1-Score Test
Logistic Regression	0.9463	0.9444	0.9465	0.9445	0.9463	0.9444	0.9464	0.9443
KNN	0.8580	0.7222	0.8608	0.7401	0.8580	0.7222	0.8588	0.7279
Random Forest	1.0000	0.8889	1.0000	0.8886	1.0000	0.8889	1.0000	0.8885
Decision Tree	1.0000	0.8712	1.0000	0.8722	1.0000	0.8712	1.0000	0.8715
SVM	0.9545	0.9066	0.9549	0.9084	0.9545	0.9066	0.9546	0.9070
XGBoost	1.0000	0.9167	1.0000	0.9168	1.0000	0.9167	1.0000	0.9164

Table VII.1: Model Performance Metrics

1. **Logistic Regression:** High test accuracy ($\sim$ 94.44%), indicating good generalization.
2. **KNN:** Moderate performance on the test set ($\sim$ 72.22%), likely due to its sensitivity to the choice of n_neighbors.
3. **Random Forest:** Overfitting on the training set (100% accuracy) but reduced generalization ($\sim$ 87.88% test accuracy).
4. **Decision Tree:** Overfitting on the training set (100% accuracy) but reduced generalization ($\sim$ 86.11% test accuracy).
5. **SVM:** Consistently high performance with test accuracy ($\sim$ 90.66%).
6. **XGBoost:** Consistently high performance with test accuracy ($\sim$ 91.67%).

As we can see there are overfitting issues with some models. We used `GridSearchCV` to tune the hyperparameters to get better performance and got the following results:

Model	Accuracy Train	Precision Train	Recall Train	F1-Score Train	Accuracy Test	Precision Test	Recall Test	F1-Score Test
Logistic Regression	0.9457	0.9458	0.9457	0.9457	0.9495	0.9492	0.9495	0.9493
KNN	0.8592	0.8615	0.8592	0.8596	0.7626	0.7790	0.7626	0.7670
Random Forest	1.0000	1.0000	1.0000	1.0000	0.8889	0.8906	0.8889	0.8889
Decision Tree	0.9968	0.9968	0.9968	0.9968	0.8712	0.8718	0.8712	0.8710
SVM	0.9476	0.9477	0.9476	0.9476	0.9520	0.9518	0.9520	0.9518
XGBoost	0.9962	0.9962	0.9962	0.9962	0.9192	0.9196	0.9192	0.9190

Table VII.2: Model Performance Metrics after Hyperparameter Tuning

Time time taken for training the models was as follows:

Model	Time Taken (seconds)
Logistic Regression	0.008992
KNN	0.001998
Random Forest	0.262442
Decision Tree	0.007005
SVM	0.237206
XGBoost	0.211886

Table VII.3: Time Taken for Model Training

As we can see Logistic Regression, KNN and Decision Tree are the fastest to train while Random Forest, SVM and XGBoost take the most time. This is expected as Random Forest and SVM are more complex models and require more time to train.

VII.3 Comparison between different Models

Comparing the matrices form Table VII.1 amd Table VII.2 we can infer:

- **Logistic Regression:** The performance metrics remained consistent after hyperparameter tuning, indicating that the model was already well-optimized.

- **KNN:** There was a significant improvement in test accuracy from 0.7222 to 0.7790, as well as improvements in precision, recall, and F1-score, suggesting that hyperparameter tuning effectively addressed underfitting.
- **Random Forest:** The test accuracy slightly decreased from 0.8889 to 0.8906, but the model maintained high performance, indicating that hyperparameter tuning helped in balancing the training and test metrics.
- **Decision Tree:** The performance metrics remained relatively stable, with a slight improvement in test precision and recall, indicating that hyperparameter tuning helped in reducing overfitting.
- **SVM:** The test accuracy improved from 0.9066 to 0.9518, along with improvements in precision, recall, and F1-score, suggesting that hyperparameter tuning significantly enhanced the model's performance.
- **XGBoost:** The test accuracy improved from 0.9167 to 0.9196, with consistent improvements in other metrics, indicating that hyperparameter tuning helped in maintaining high performance and efficient learning.

SECTION VIII

DISCUSSION

VIII.1 Analysis of Results

Logistic Regression performed reliably, especially after minor improvements post-tuning. **KNN** and **Decision Tree** struggled with certain classes, as evident from their lower Precision-Recall curves. **XGBoost** and **SVM** consistently achieved high F1-scores, indicating a strong balance between precision and recall.

Random Forest, **Decision Tree**, and **XGBoost** are inherently complex and hence prone to overfitting. **GridSearchCV** significantly improved the balance between training and test metrics for most models, particularly **KNN** and **SVM**.

VIII.2 Anomalies and Unexpected Findings

The dataset we chose did not have any class imbalances but there may have been challenges related to it, which can affect model predictions, particularly in underrepresented categories.
Also we found that among the 19 features, 13 were not much important such as 3G, 4G, front and primary camera quality, and screen resolution components. Removing redundant features improved model performance and reduced complexity.

VIII.3 Limitations

- If certain price categories were underrepresented, the model might not perform equally well across all classes, potentially favoring more common categories.
- The impact of feature selection and engineering steps could vary, and further refinement might be needed to capture more complex relationships in the data.
- The results might not generalize well to unseen data if there were biases in the dataset or if the training set wasn't diverse enough.

SECTION IX

CONCLUSION

IX.1 Key Takeaways

The key takeaways from this project are:

- The project successfully developed a classification model capable of categorizing smartphones into distinct price segments (Low, Medium, High, Very High) using a variety of machine learning algorithms.
- A thorough feature analysis revealed the most influential factors for predicting smartphone prices, such as RAM, battery power, and camera quality.
- The systematic feature selection process, including normalization and encoding, highlighted critical features that contributed significantly to model accuracy.
- Effective feature engineering, including normalization of screen and pixel areas, ensured a refined and optimized dataset for training.
- Various machine learning models were implemented, tuned, and evaluated. Metrics such as accuracy, precision, and recall were used to compare the models.
- The project provided clear insights into which algorithms were more effective for this classification task. For example, ensemble methods like the Voting Classifier generally outperformed individual classifiers.
- The utilization of metrics like confusion matrices, ROC curves, and AUC scores provided a comprehensive understanding of the model's accuracy and reliability.

- The project made methodological contributions by developing a replicable framework for smartphone price categorization, emphasizing feature analysis and multi-algorithm comparison.
- The workflow established in this project serves as a template for future predictive modeling challenges, particularly in consumer market analysis using machine learning techniques.

IX.2 Reflection on Objectives

1. **Robust Classification Model :**
   - The project met its objective of building an accurate classification model, using multiple algorithms and optimizing them for high predictive accuracy.
   - The most significant features influencing smartphone pricing were identified, meeting the goal of leveraging key variables for prediction.
2. **Feature Analysis and Selection :**
   - A comprehensive feature analysis was performed, and a systematic approach to feature selection was developed, successfully identifying the most influential smartphone attributes.
   - The selection of relevant features directly impacted the model's performance, aligning with the project's goals.
3. **Comparative Algorithm Performance :**
   - Multiple machine learning algorithms were tested, and their performances were compared, allowing for the selection of the most effective model for smartphone categorization.
   - The comparison revealed insights into which algorithms handled the dataset's characteristics best, achieving the project's objective of comparative analysis.
4. **Practical Applicability :**
   - The practical utility of the model was demonstrated, showcasing how it could assist both manufacturers and consumers in making data-driven decisions, achieving this core objective.
5. **Methodological Contribution :**
   - A replicable methodology was established, contributing to market analysis using machine learning. The project's systematic approach, from feature analysis to model comparison, stands as a reference for similar challenges.

IX.3 Future Work

1. **Addressing Class Imbalances:**
   - Future research could focus on balancing the dataset more effectively, employing techniques like SMOTE

(Synthetic Minority Over-sampling Technique) to handle underrepresented categories.

- Experimenting with different sampling strategies or cost-sensitive algorithms might yield better performance across all price segments.

2. **Advanced Feature Engineering:**
   - Introducing more sophisticated feature engineering methods, such as polynomial features or interaction terms, might capture deeper relationships between variables.
   - Incorporating external data sources like user reviews or brand reputation could provide additional features for enhanced model predictions.
3. **Deep Learning Approaches:**
   - Exploring deep learning models, such as neural networks, could help capture complex patterns in the data that traditional machine learning models might miss.
   - Fine-tuning deep learning algorithms might lead to better generalization and improved accuracy for smartphone price prediction.
4. **Model Interpretability: :**
   - Enhancing model interpretability by using techniques like SHAP (Shapley Additive Explanations) or LIME (Local Interpretable Model-agnostic Explanations) can provide more insights into how predictions are made, especially for complex models.
5. **Expanding Practical Use Cases:**
   - Future work could extend beyond categorization, such as predicting the price value itself or identifying which features drive consumer satisfaction.
   - The developed model can be adapted to other product categories, providing a broader application in market analysis and consumer research.

REFERENCES

[1] Géron, Aurélien. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 3rd ed., O'Reilly Media, 2023.

[2] Asim, Muhammad, and Zafar Khan. "Mobile Price Class Prediction Using Machine Learning Techniques." *International Journal of Computer Applications*, March 2018. DOI: 10.5120/ijca2018916555.

[3] Chandrashekhara, K. T., and M. Thungamani. "Smartphone Price Prediction in Retail Industry Using Machine Learning Techniques." In *Emerging Research in Electronics, Computer Science and Technology, Lecture Notes in Electrical Engineering 545*, edited by V. Sridhar et al., Springer Nature Singapore Pte Ltd., 2019. DOI: 10.1007/978-981-13-5802-9_34.

[4] Ercan & Şimşek, "Mobile Phone Price Classification Using Machine Learning." *International Journal of Advanced Natural Sciences and Engineering Researches*, vol. 7, no. 4, 2023, pp. 458-462. DOI: 10.59287/ijanser.791.

[5] Abbasi, Muhammad Hasnain, Abdul Sajid, Muhammad Arshad Awan, and Ayeb Amani. "Predicting The Price Of Used Electronic Devices Using Machine Learning Techniques." *International Journal of Computing and Related Technologies*, vol. 4, no. 1, 2024. Available at: https://www.researchgate.net/publication/377526585.

[6] Gupta, Akash, and Suhasini Kottur. "Mobile Price Prediction by Its Features Using Predictive Model of Machine Learning." *UGC Care Journal*, August 2020. DOI: 10.13140/RG.2.2.20054.52800.

[7] Kumuda S, Vishal Karur, and Karthick Balaje S. E. "Prediction of Mobile Model Price Using Machine Learning Techniques." *International Journal of Engineering and Advanced Technology (IJEAT)*, vol. 11, no. 1, October 2021. DOI: 10.35940/ijeat.A3219.1011121.