Mobile Price Classification Using Decision Tree Classifier

Abstract

This report presents a comprehensive analysis of a machine learning project aimed at classifying mobile phones into price categories based on their specifications. The dataset comprises various features, including battery power, RAM, and screen resolution etc. A Decision Tree Classifier was employed to build the predictive model. Key aspects of the project include data preprocessing, noise treatment using K-Nearest Neighbors (KNN) Imputer, feature selection, hyperparameter tuning, and model evaluation. The final model achieved an accuracy of approximately 89% on the test data, demonstrating its effectiveness in handling noisy data and making accurate predictions.

Introduction

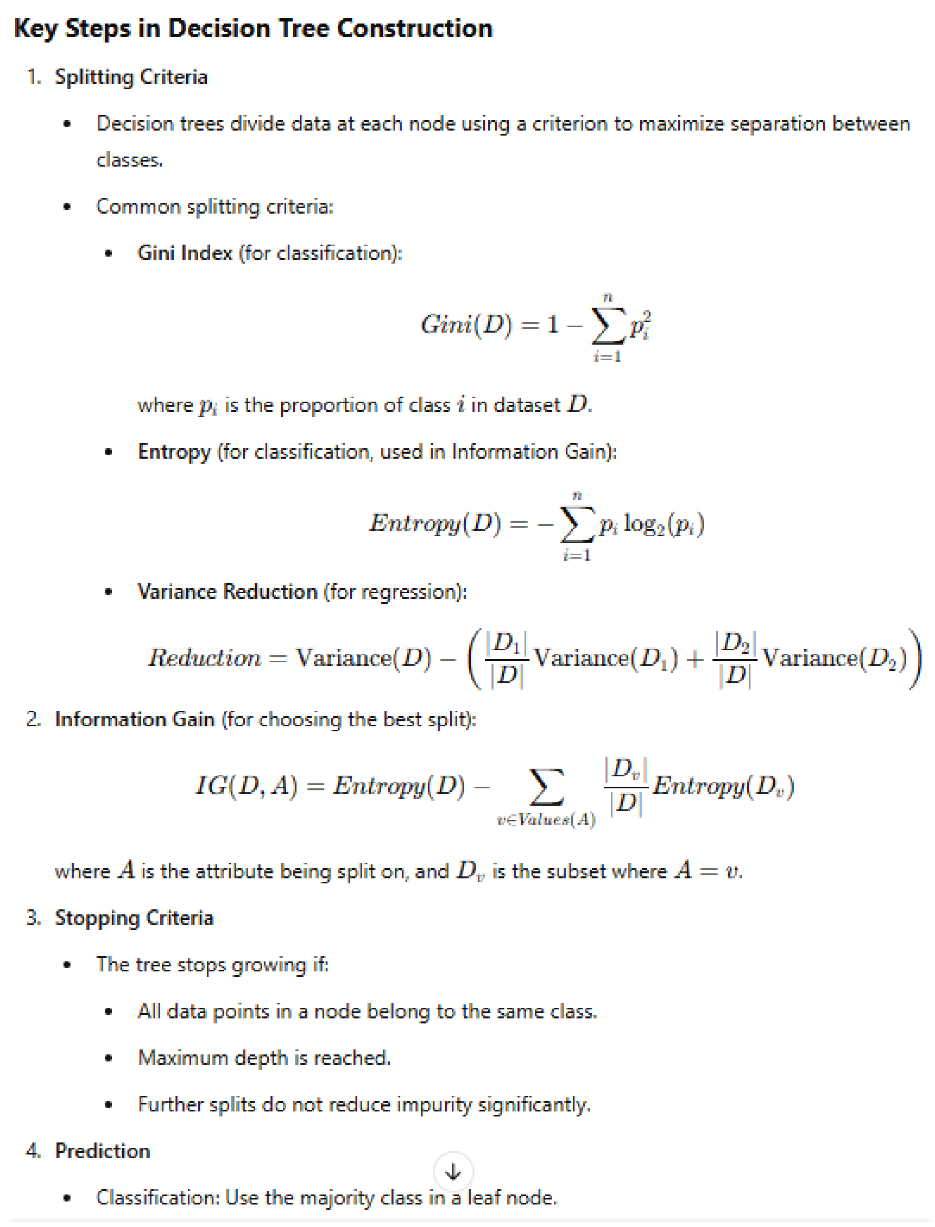
In the competitive mobile phone market, accurately predicting the price range of a device based on its specifications is valuable for manufacturers and consumers. This project utilizes a dataset containing various features of mobile phones to classify them into different price categories. Given the potential presence of noise and irrelevant features, the project emphasizes robust data preprocessing and model optimization to enhance predictive performance.

Data Preprocessing and Noise Treatment

Data preprocessing is a critical step in machine learning, especially when dealing with real-world datasets that may contain noise or missing values. In this project, features such as pixel height (px_height) and screen width (sc_w) exhibited significant noise, which could adversely affect the performance of the Decision Tree model. To address this, the K-Nearest Neighbors (KNN) Imputer was employed. The KNN Imputer replaces missing or noisy values by averaging the values of the nearest neighbors, determined based on Euclidean distance. In this case, the imputer was configured with k=2, meaning it considered the two nearest neighbors for imputation. This approach helps preserve the underlying data structure while mitigating the impact of noise.

Decision Tree Classifier

A Decision Tree is a supervised learning algorithm used for both classification and regression tasks. It splits the data into subsets based on the value of input features, resulting in a tree-like model of decisions. Each internal node represents a "test" on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label. The paths from the root to the leaf represent classification rules. Decision Trees are easy to interpret and can handle both numerical and categorical data. However, they are prone to overfitting, especially when dealing with noisy data, which necessitates careful tuning of hyperparameters.



Key Steps in Decision Tree Construction

1. **Splitting Criteria**
   - Decision trees divide data at each node using a criterion to maximize separation between classes.
   - Common splitting criteria:
     - **Gini Index** (for classification):

$$Gini(D) = 1 - \sum_{i=1}^{n} p_i^2$$

       where p_i is the proportion of class i in dataset D.
     - **Entropy** (for classification, used in Information Gain):

$$Entropy(D) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

     - **Variance Reduction** (for regression):

$$Reduction = \text{Variance}(D) - \left(\frac{|D_1|}{|D|}\text{Variance}(D_1) + \frac{|D_2|}{|D|}\text{Variance}(D_2)\right)$$

2. **Information Gain** (for choosing the best split):

$$IG(D, A) = Entropy(D) - \sum_{v \in Values(A)} \frac{|D_v|}{|D|} Entropy(D_v)$$

   where A is the attribute being split on, and D_v is the subset where $A = v$.
3. **Stopping Criteria**
   - The tree stops growing if:
     - All data points in a node belong to the same class.
     - Maximum depth is reached.
     - Further splits do not reduce impurity significantly.
4. **Prediction**
   - Classification: Use the majority class in a leaf node.

Feature Selection

Feature selection involves identifying and selecting the most relevant features for use in model construction. This process helps in reducing the dimensionality of the data, improving model performance, and reducing overfitting. In this project, the Drop-Column Feature Importance method was utilized. This technique involves training the model with all features and then iteratively removing each feature to observe the impact on model performance. Features that, when removed, result in negligible or negative changes in performance are considered less important and can be excluded from the model. The selected features in this project were ram, battery_power, px_width, and px_height, as they showed the most significant impact on predicting the price range.

Hyperparameter Tuning

Hyperparameter tuning is the process of optimizing the parameters that govern the training process of a model to improve its performance. For the Decision Tree Classifier, several hyperparameters were tuned:

- **Criterion:** This parameter determines the function to measure the quality of a split. The options are 'gini' for the Gini impurity and 'entropy' for the information gain. Both criteria were evaluated to determine which provided better performance.
- **Max Depth (max_depth):** This parameter sets the maximum depth of the tree. Limiting the depth helps prevent the model from learning noise in the data, thereby reducing overfitting.
- **Minimum Samples Split (min_samples_split):** This parameter specifies the minimum number of samples required to split an internal node. Setting this parameter helps ensure that nodes have enough data to make meaningful splits.
- **Minimum Samples Leaf (min_samples_leaf):** This parameter sets the minimum number of samples required to be at a leaf node. It helps prevent the model from creating nodes that represent outliers or noise.

Grid Search Cross-Validation was employed to systematically explore combinations of these hyperparameters and select the set that resulted in the best model performance.

Model Evaluation

The final Decision Tree model, with the optimized hyperparameters, was evaluated on both training and test datasets. The model achieved an accuracy of approximately 96% on the training data and 89% on the test data. The slight decrease in accuracy on the test data suggests a mild overfitting, which was addressed by adjusting the max_depth parameter to balance bias and variance.

Conclusion

This project successfully developed a noise-resilient Decision Tree Classifier for predicting mobile phone price ranges based on key specifications. By effectively handling noisy data with the KNN Imputer, selecting relevant features, and tuning hyperparameters, the model achieved high accuracy while maintaining simplicity and interpretability. Future work could explore ensemble methods, such as Random Forests or Gradient Boosting, to further enhance model performance and robustness.

Reference links:

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

https://colab.research.google.com/drive/1zJmt8uqkWcGz5PkgWrSvyNQ_3hA9Lrls?usp=sharing