Mobile price Prediction

Problem Description :

Problem Description:

In the Mobile Price Prediction project, we aim to build a machine learning model that can predict the price range of mobile phones based on various features and specifications. The goal is to develop a predictive tool that can assist buyers, sellers, and manufacturers in making informed decisions related to mobile phone prices.

Background Information:

Mobile phones have become an essential part of modern life, and the mobile phone market is constantly evolving with new models being introduced regularly. However, with such a vast array of mobile phones available in the market, it can be challenging for consumers to decide which phone to buy, especially with respect to their budget preferences.

Price is a crucial factor for both consumers and manufacturers. For consumers, the price of a mobile phone often determines their purchasing decision, and they look for the best value for their money. On the other hand, manufacturers need to competitively price their products to stay competitive in the market and meet the demands of various customer segments.

This is where predictive models can play a significant role. By analyzing historical data on mobile phone specifications and prices, we can train a machine learning model to predict the price range of a mobile phone based on its features. This prediction can help consumers find mobile phones within their budget and enable manufacturers to set competitive prices for their products.

Dataset Information:

The dataset used in this project contains information about various mobile phone specifications and their corresponding price ranges. Each data entry represents a different mobile phone, and the features include:

1. **battery_power:** Total energy a battery can store in mAh.
2. **blue:** Bluetooth support (1 if supported, 0 if not supported).
3. **clock_speed:** Speed at which the microprocessor executes instructions.
4. **dual_sim:** Dual SIM support (1 if supported, 0 if not supported).
5. **fc:** Front Camera mega pixels.
6. **four_g:** 4G support (1 if supported, 0 if not supported).
7. **int_memory:** Internal Memory in GB.
8. **m_dep:** Mobile Depth in cm.
9. **mobile_wt:** Weight of the mobile phone.
10. **n_cores:** Number of cores of the processor.
11. **pc:** Primary Camera mega pixels.
12. **px_height:** Pixel Resolution Height.
13. **px_width:** Pixel Resolution Width.
14. **ram:** Random Access Memory (RAM) in MB.
15. **sc_h:** Screen Height of the mobile in cm.
16. **sc_w:** Screen Width of the mobile in cm.
17. **talk_time:** The longest time that a single battery charge will last when you are constantly talking on the phone in minutes.
18. **three_g:** 3G support (1 if supported, 0 if not supported).
19. **touch_screen:** Touchscreen support (1 if supported, 0 if not supported).
20. **wifi:** Wifi support (1 if supported, 0 if not supported).
21. **price_range:** Target variable representing the price range of the mobile phone (0: low cost, 1: medium cost, 2: high cost, 3: very high cost).

The dataset provides a diverse range of features, each contributing to the overall specifications and pricing of mobile phones. The price_range is the target variable that we want to predict based on the other features.

Project Objective:

The objective of this project is to build a predictive model that can accurately classify the price range of mobile phones based on their specifications. By doing so, we aim to create a tool that can assist users in making informed decisions about mobile phone purchases and help manufacturers competitively price their products in the market.

Possible Framework :

1. **Importing Libraries:** Begin by importing the necessary Python libraries such as pandas, numpy, matplotlib, seaborn, and scikit-learn. These libraries are essential for data manipulation, visualization, and machine learning tasks.
2. **Data Loading and Exploration:**

- Load the dataset containing mobile phone specifications and price ranges using pd.read_csv().
- Use dataset.head() to display the first few rows of the dataset to get an overview of the data.
- Check the data types and null values using dataset.info() and dataset.isnull().sum().

3. **Data Visualization:**

- Create visualizations to understand the distribution of the target variable (price_range) and its relationship with other features.
- Use scatter plots, pair plots, and joint plots to explore the relationships between numerical features and the target variable.
- Create bar plots and pie charts to visualize categorical features.

4. **Data Preprocessing:**

- Prepare the data for training the machine learning models.
- Separate the features (X) and the target variable (y).
- Convert categorical variables to numeric using techniques like label encoding or one-hot encoding.
- Split the dataset into training and testing sets using train_test_split().

5. **Model Building:**

- Choose the appropriate machine learning algorithm for this classification problem. Common choices include Linear Regression, K-Nearest Neighbors (KNN), Logistic Regression, Decision Tree, and Random Forest.
- Build and train the selected model on the training data using the respective algorithm's functions (fit()).

6. **Model Evaluation:**

- Evaluate the performance of the trained model on the test data using score() or other appropriate evaluation metrics like accuracy, precision, recall, and F1-score.
- If using KNN, tune the hyperparameter n_neighbors by plotting the error rate vs. K value and selecting the optimal K with the lowest error rate.

7. **Feature Importance (Optional):**
- If using decision tree or random forest, visualize the feature importance ranking to understand which features are most influential in predicting the price range.
- Create a bar plot to show the importance of each feature.

8. **Predicting Mobile Prices:**
- After building and evaluating the model, use it to predict mobile phone prices on a new dataset.
- Load the new dataset using pd.read_csv() and preprocess it similar to the training dataset.
- Use the trained model to predict the price range for each mobile phone in the new dataset.

9. **Model Comparison (Optional):**
- Compare the performance of different models used during training and select the best-performing one based on the evaluation metrics.
- Final Visualization (Optional):
- Visualize the actual prices vs. predicted prices to observe how well the model has performed.
- Plot the scatter plot of actual prices against predicted prices to visually compare them.

10. **Conclusion:**
- Summarize the findings and insights from the project, including the predictive performance of the model.
- Discuss the model's ability to predict mobile prices accurately and any limitations or improvements that can be made.

11. **Future Work:**
- Provide recommendations for future work to enhance the model's performance or expand the project to include additional features or data sources.

12. **References (Optional):**
- If any external resources or research papers were used during the project, list them as references.

Code Explanation :

*If this section is empty, the explanation is provided in the .ipynb file itself.

1. **Importing Libraries:**

- In this step, the necessary Python libraries are imported to perform various tasks throughout the code. These libraries include pandas, numpy, matplotlib, seaborn, and scikit-learn.

2. **Data Loading and Exploration:**

- The code starts by loading the dataset containing mobile phone specifications and price ranges using the pd.read_csv() function from pandas.
- The dataset.head() function is used to display the first few rows of the dataset to get an overview of the data.
- The dataset.info() function is used to check the data types and the number of non-null values in each column.
- The dataset.describe() function provides statistical summary of the numerical columns in the dataset, such as mean, standard deviation, minimum, maximum, and quartiles.

3. **Data Visualization:**

- In this step, various data visualization techniques are used to better understand the data and its relationships.
- sns.pairplot() creates a pair plot, which is a grid of scatter plots, showing the relationships between all pairs of numerical features. The hue parameter is used to differentiate the data points based on the 'price_range' class.
- sns.jointplot() creates a joint plot showing the distribution of 'ram' (random access memory) and its relationship with 'price_range' using a kernel density estimation (KDE) curve.
- sns.pointplot() creates a point plot showing the average 'int_memory' (internal memory) for each price range category.
- Pie charts are created to visualize the distribution of binary features 'three_g' and 'four_g'.

4. **Data Preprocessing:**

- Before building machine learning models, the data needs to be prepared and preprocessed.

- The features (independent variables) are stored in 'X', and the target variable (dependent variable) is stored in 'y'.
- Categorical variables are converted to numeric using techniques like label encoding or one-hot encoding.
- The dataset is split into training and testing sets using train_test_split() from scikit-learn.

5. **Model Building:**

- In this step, different machine learning models are built and trained on the training data to predict the 'price_range'.
- The models used in the code are Linear Regression, K-Nearest Neighbors (KNN), Logistic Regression, Decision Tree, and Random Forest.
- Each model is created using its respective class (e.g., LinearRegression, KNeighborsClassifier, LogisticRegression, DecisionTreeClassifier, RandomForestClassifier) and then trained on the training data using the fit() method.

6. **Model Evaluation:**

- After training the models, their performance is evaluated on the testing data using score() or other evaluation metrics like accuracy, precision, recall, and F1-score.
- For KNN, the code also tunes the hyperparameter 'n_neighbors' by plotting the error rate vs. K value to find the optimal K with the lowest error rate.

7. **Feature Importance (Optional):**

- For decision tree and random forest models, the code visualizes the feature importance ranking to identify the most influential features in predicting the price range.
- A bar plot is created to show the importance of each feature.

8. **Predicting Mobile Prices:**

- After building and evaluating the models, the code uses the trained models to predict the price range for mobile phones in a new dataset ('data_test').
- The new dataset is loaded and preprocessed similar to the training dataset.
- The trained models are then used to predict the price range for each mobile phone in the new dataset using the predict() method.

9. **Model Comparison (Optional):**

- The code compares the performance of different models used during training and selects the best-performing one based on evaluation metrics.

10. **Final Visualization (Optional):**
- The code creates visualizations to compare the actual prices vs. predicted prices, providing insights into how well the models have performed.

11. **Conclusion:**
- The project concludes by summarizing the findings and insights from the project, including the predictive performance of the models and any limitations or improvements that can be made.

12. **Future Work:**
- The future work section provides recommendations for enhancing the model's performance or expanding the project to include additional features or data sources.

Overall, the code follows a standard machine learning workflow, including data loading, exploration, visualization, preprocessing, model building, evaluation, and prediction. The goal of the project is to predict mobile prices based on their specifications, and the code demonstrates various machine learning algorithms to achieve this task.

Future Work :

Predicting mobile prices based on their specifications is an interesting task, and there are several ways to further improve the project. Below are the steps for future work and how to implement them:

1. Data Augmentation:

- One way to improve the model's performance is to augment the existing dataset with more data. This can be done by collecting additional data from various sources or generating synthetic data using techniques like data augmentation.

2. Feature Engineering:

- Feature engineering involves creating new features or transforming existing features to improve model performance. For example, we can create new features like 'screen-to-body ratio' or 'battery capacity per weight' that may have better predictive power.

3. Hyperparameter Tuning:

- The performance of machine learning models can be enhanced by fine-tuning their hyperparameters. Implement grid search or random search techniques to find the best hyperparameters for the models.

4. Ensemble Methods:

- Implement ensemble methods like stacking or boosting to combine the predictions of multiple models and improve overall performance.

5. Different Algorithms:

- Try different machine learning algorithms that are suitable for regression tasks, such as Support Vector Machines (SVM), Gradient Boosting, and Neural Networks.

6. Cross-Validation:

- Implement k-fold cross-validation to get a more robust estimate of the model's performance and reduce the risk of overfitting.

7. Feature Importance Analysis:

- Conduct an in-depth analysis of feature importance to identify the most relevant features that contribute significantly to the mobile price prediction. Remove less important features to simplify the model.

8. Handling Outliers:

- Check for outliers in the dataset and apply appropriate techniques (e.g., removing outliers or transforming them) to improve model performance.

9. Model Interpretability:

- Implement techniques to make the models more interpretable, especially for decision tree-based models. This will help understand how specific features influence the predictions.

10. Advanced Visualization:

- Create more advanced visualizations, such as 3D plots or interactive visualizations, to gain deeper insights into the relationships between features and the target variable.

Step-by-Step Guide to Implement Future Work:

1. **Data Collection and Augmentation:**
- Collect additional data from different sources or use data augmentation techniques to increase the size and diversity of the dataset.
2. **Feature Engineering:**
- Analyze the existing features and create new relevant features that may improve the model's predictive power.
3. **Hyperparameter Tuning:**
- Implement grid search or random search to find the best hyperparameters for the machine learning models.
4. **Ensemble Methods:**
- Combine the predictions of multiple models using techniques like stacking or boosting.
5. **Try Different Algorithms:**

- Experiment with different machine learning algorithms to identify the ones that work best for the mobile price prediction task.

6. **Cross-Validation:**
- Implement k-fold cross-validation to evaluate the model's performance more effectively.
7. **Feature Importance Analysis:**
- Use techniques like permutation importance or SHAP (SHapley Additive exPlanations) to analyze feature importance.
8. **Handling Outliers:**
- Identify and handle outliers in the dataset appropriately.
9. **Model Interpretability:**
- Explore techniques like partial dependence plots or SHAP values to interpret the models.
10. **Advanced Visualization:**
- Create more advanced visualizations to gain deeper insights into the data and model behavior.

By implementing these future work steps, the mobile price prediction project can be enhanced, and the model's performance can be improved, resulting in more accurate and reliable predictions for mobile prices.

Concept Explanation :

K-Nearest Neighbors (KNN) Algorithm: A Friendly Guide

Hey there, friend! Today, we're going to explore the magical world of the K-Nearest Neighbors (KNN) algorithm! Don't worry; it's not as complicated as it sounds. In fact, KNN is like having friendly neighbors who help you decide which category a new friend belongs to based on the friends living closest to them. Let's dive in!

Imagine a World of Friends:

Let's say we have a group of friends, and we know whether they like high-priced mobiles (class A) or low-priced mobiles (class B). We also know two important characteristics of each friend: "RAM" and "Battery Power." We have plotted these friends on a graph, where "RAM" is on the x-axis, and "Battery Power" is on the y-axis.

The Magical KNN Process:

Now, let's welcome a new friend to the group! This new friend hasn't told us whether they like high-priced or low-priced mobiles, but we know their "RAM" and "Battery Power." How do we determine which category they belong to?

Here comes the magical KNN process to the rescue! KNN stands for "K-Nearest Neighbors," where "K" represents the number of friends living closest to the new friend.

Step 1: Find the K Nearest Neighbors:

First, we look at the "K" closest friends (neighbors) to our new friend based on their "RAM" and "Battery Power" values. These friends have similar characteristics and might have similar preferences when it comes to mobiles.

Step 2: Majority Rules:

Now, let's see how these "K" neighbors like their mobiles. If most of them have high-priced mobiles, we will predict that our new friend also likes high-priced mobiles (class A). If most of them have low-priced mobiles, we'll predict that our new friend prefers low-priced mobiles (class B).

Time to Make Friends:

Isn't this cool? KNN helps us make friends with the right crowd based on the friends living closest to us on the graph. It's like having a group of friends who give us recommendations for things we might like based on what they like!

Let's Play with Different K Values:

Oh, wait! The "K" in KNN is not fixed. We can try different values of "K" to see which one gives the best predictions. For example, if we set K=3, we'll look at the three closest neighbors. But if we set K=5, we'll consider the five closest neighbors instead. So, it's like hanging out with different groups of friends and getting different advice!

KNN's Friendly Pros and Cons:

Pros: KNN is simple to understand, doesn't need much training (learning), and can work well for small datasets.

Cons: It might not perform well with large datasets, as it requires looking at all the neighbors for every new friend. Also, it's sensitive to outliers (friends with unusual preferences), which can sometimes lead to strange predictions.

Conclusion:

So there you have it! KNN is like your friendly neighborhood algorithm that helps you predict the category of a new friend based on the preferences of the friends living closest to them. It's easy to make friends with KNN, and it's fun to experiment with different groups of friends (K values) to find the best predictions.

Now, you're all set to use KNN for mobile price predictions and make friends with the right crowd in the world of machine learning! Happy predicting!

Exercise Questions :

1. **What is the purpose of the "Mobile Price Prediction" project, and what is the dataset used for this analysis?**

Answer: The purpose of the "Mobile Price Prediction" project is to predict the price range of mobile phones based on various features like RAM, battery power, camera specifications, etc. The dataset used for this analysis contains information about different mobile phones, along with their corresponding price ranges.

2. **Can you explain the process of data visualization used in this project?**

Answer: Data visualization is the process of representing data in graphical or pictorial form to gain insights. In this project, various types of plots like pair plots, joint plots, bar plots, box plots, and scatter plots are used to understand the relationships between different features and the price range of mobile phones.

3. **How does the K-Nearest Neighbors (KNN) algorithm work in this project?**

Answer: The KNN algorithm works by finding the "K" nearest neighbors to a new data point based on the feature values. It then predicts the category of the new data point by taking a majority vote from the categories of its "K" neighbors. In this project, KNN is used to predict the price range of mobile phones based on the characteristics of similar mobile phones in the dataset.

4. **Why is it important to split the dataset into training and testing sets?**

Answer: Splitting the dataset into training and testing sets is essential to evaluate the performance of the machine learning model accurately. The training set is used to train the model, while the testing set is used to assess how well the model generalizes to unseen data. This prevents overfitting, where the model performs well on the training data but poorly on new data.

5. **How is the error rate used to determine the optimal value of "K" in the KNN algorithm?**

Answer: The error rate is calculated for different values of "K" in the KNN algorithm. It represents the proportion of misclassifications on the testing set. The optimal value of "K" is chosen where the error rate is the lowest, as it indicates the best balance between overfitting and underfitting.

6. **What are the advantages and disadvantages of the KNN algorithm?**

Answer: The advantages of the KNN algorithm include its simplicity, ease of implementation, and ability to work well with small datasets. However, its disadvantages include sensitivity to outliers and inefficiency with large datasets, as it needs to examine all data points for every prediction.

7. **How is the Random Forest algorithm different from the KNN algorithm?**

Answer: The Random Forest algorithm is an ensemble method that builds multiple decision trees and combines their predictions. In contrast, KNN is a lazy learning algorithm that stores all the training data points and makes predictions based on the proximity of the new data point to existing data points. Random Forest is better suited for large datasets and can handle a higher number of features efficiently.

8. **What is the purpose of the PolynomialFeatures in this project, and how does it improve the model's performance?**

Answer: PolynomialFeatures is used to create polynomial features from the existing features in the dataset. It transforms the original features into higher-degree features, allowing the model to capture non-linear relationships between the features and the target variable. This can improve the model's performance and accuracy.

9. **How is the accuracy of the machine learning models evaluated in this project?**

Answer: The accuracy of the machine learning models (Linear Regression, KNN, Logistic Regression, Decision Tree, and Random Forest) is evaluated using the "score" method, which calculates the accuracy of the model on the testing set. It represents the proportion of correct predictions made by the model.

10. **What are the potential areas of improvement for this "Mobile Price Prediction" project?**

Answer: Some potential areas of improvement for this project include trying different machine learning algorithms, fine-tuning hyperparameters, handling missing data more effectively, and exploring feature engineering techniques to extract more meaningful information from the existing features. Additionally, gathering more diverse and extensive data could further enhance the model's accuracy and predictive power.