**Predicting Units Sold with Decision Tree Regression**

**Introduction:**

The provided code implements a Decision Tree Regressor model to predict the number of units sold based on features like "Total Price" and "Base Price". This type of model is commonly used in regression tasks, where the goal is to predict a continuous target variable.

**Code Explanation:**

**Data Splitting**

The first step in the code involves splitting the data into training and testing sets. This is crucial to evaluate how well the model generalizes to unseen data. The data is divided into two sets:

X: Contains the features used for prediction.

y: Contains the target variable (number of units sold).

test\_size=0.2: Specifies that 20% of the data will be used for testing, and the remaining 80% for training.

random\_state=42: Ensures that the data is split in a reproducible manner.

**Code:**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**Model Training:**

Next, a Decision Tree Regressor model is created and trained using the training data:

DecisionTreeRegressor(): Initializes a Decision Tree Regressor model.

model.fit(X\_train, y\_train): Fits the model to the training data, allowing it to learn the relationships between the features and the target variable.

**Code:**

model = DecisionTreeRegressor()

model.fit(X\_train, y\_train)

**Model Evaluation:**

The code then calculates the R-squared scores to assess the model's performance:

model.score(X\_train, y\_train): Computes the R-squared score on the training data, indicating how well the model fits the training set.

model.score(X\_test, y\_test): Calculates the R-squared score on the testing data, providing an evaluation of the model's generalization ability.

**Code:**

train\_score = model.score(X\_train, y\_train)

test\_score = model.score(X\_test, y\_test)

**Making a Prediction:**

After training, the model is used to make a prediction based on specific feature values:

features: Contains the feature values for which a prediction is desired.

model.predict(features): Predicts the target variable (number of units sold) based on the provided features.

**Code:**

features = np.array([[133.00, 140.00]])

prediction = model.predict(features)

**Printing Results:**

Finally, the code prints out the R-squared scores and the predicted number of units sold:

The R-squared scores provide an indication of how well the model is performing on both the training and testing sets.

The predicted number of units sold is displayed for the specified "Total Price" and "Base Price".

**Code:**

print("\nR-squared scores for Train and Test:")

print(f"Training R-squared score: {train\_score}")

print(f"Testing R-squared score: {test\_score}")

print(f"\nPredicted Units Sold for Total Price={Total Price} and Base Price={Base Price}: {prediction[0]}")

**Output:**

**Conclusion:**

The provided code efficiently implements a Decision Tree Regressor model for predicting the number of units sold. By splitting the data into training and testing sets, training the model, and evaluating its performance, this code serves as a foundation for regression tasks involving similar datasets. The resulting model can be used to make accurate predictions based on input features.

Top of Form