## Description

This repository contains a Python code example for a machine learning task using various regression algorithms. The code is implemented using the scikit-learn library and includes the following steps:

1. \*\*Data Loading and Preprocessing\*\*: The code starts by loading the "auto-mpg" dataset, which contains information about automobiles. It preprocesses the data by handling missing values and splitting it into features (X) and the target variable (y).

2. \*\*Data Splitting\*\*: The dataset is split into training and testing sets using the `train\_test\_split` function. Standardization is also applied to the features using `StandardScaler`.

3. \*\*Support Vector Regression (SVR)\*\*: Support Vector Regression with a linear kernel and regularization parameter C=0.1 is used to build a regression model. The model's performance is evaluated using the R-squared score.

4. \*\*Hyperparameter Tuning for SVR\*\*: Grid search is performed to find the best hyperparameters for SVR. Cross-validation is used for hyperparameter tuning, and the best model is selected. The model's performance is evaluated again.

5. \*\*Decision Tree Regression\*\*: A decision tree regression model with a specified maximum depth and minimum samples split is trained and evaluated.

6. \*\*Hyperparameter Tuning for Decision Tree\*\*: Grid search is again applied to optimize the hyperparameters of the decision tree model. The best model is selected based on cross-validation results, and its performance is evaluated.

7. \*\*Visualization\*\*: The code includes a visualization of the trained decision tree model using the `tree.plot\_tree` function.

8. \*\*K-Nearest Neighbors (KNN) Regression\*\*: A K-Nearest Neighbors regression model with a specified number of neighbors is trained and evaluated.

9. \*\*Hyperparameter Tuning for KNN\*\*: Grid search is performed to find the optimal number of neighbors and weight function for KNN. The best model is selected based on cross-validation results, and its performance is evaluated.

10. \*\*Evaluation Metrics\*\*: The code calculates and reports metrics such as R-squared score and accuracy score for each regression model.

This code provides a comprehensive example of how to perform regression tasks, including data preprocessing, model training, hyperparameter tuning, and evaluation, using scikit-learn. It can serve as a useful reference for anyone working on similar machine learning projects.