ASSESSMENT-3

Step 1: Problem Understanding:

We identified the goal as predicting house prices based on features like square footage, number of rooms, and location.

Step 2: Data Collection:

We gathered the dataset, which included features such as number of bedrooms, bathrooms, house size, and price as the target variable.

Step 3: Data Preprocessing:

We handled missing values, encoded categorical features, and normalized numerical data to ensure consistent scaling.

Step 4: Data Splitting:

The dataset was split into training (80%) and testing (20%) sets to train and evaluate the model.

Step 5: Model Selection:

We chose Linear Regression initially, followed by Random Forest and Gradient Boosting for better predictions.

Step 6: Model Training:

The models were trained using the training set, with hyperparameters fine-tuned to optimize accuracy.

Step 7: Model Evaluation:

We evaluated model performance using metrics like Mean Squared Error and R-squared, with Gradient Boosting performing best.

Step 8: Hyperparameter Tuning:

GridSearchCV was used to find the best hyperparameters for the models, further improving prediction accuracy.

Step 9: Conclusion:

The Gradient Boosting model achieved the highest accuracy and is ready for potential deployment.

Code:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, r2_score

# Load the dataset using space as a delimiter
data = pd.read_csv('housing.csv', sep='\s+', header=None)

#EDA

# Define column names
column_names = [
    'CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE',
    'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT', 'price'
]
data.columns = column_names

# Prepare the features and target variable
X = data.drop('price', axis=1)
y = data['price']

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 1. Linear Regression
linear_model = LinearRegression()
```

```
linear_model.fit(X_train, y_train)
y_pred_linear = linear_model.predict(X_test)
mae_linear = mean_absolute_error(y_test, y_pred_linear)
r2_linear = r2_score(y_test, y_pred_linear)

print(f"Linear Regression:\n  Mean Absolute Error: {mae_linear:.2f}\n  R² Score: {r2_linear:.2f}\
n")

# 2. Decision Tree Regression
tree_model = DecisionTreeRegressor()
tree_model.fit(X_train, y_train)
y_pred_tree = tree_model.predict(X_test)
mae_tree = mean_absolute_error(y_test, y_pred_tree)
r2_tree = r2_score(y_test, y_pred_tree)

print(f"Decision Tree Regression:\n  Mean Absolute Error: {mae_tree:.2f}\n  R² Score:
{r2_tree:.2f}\n")

# 3. Random Forest Regression
forest_model = RandomForestRegressor()
forest_model.fit(X_train, y_train)
y_pred_forest = forest_model.predict(X_test)
mae_forest = mean_absolute_error(y_test, y_pred_forest)
r2_forest = r2_score(y_test, y_pred_forest)

print(f"Random Forest Regression:\n  Mean Absolute Error: {mae_forest:.2f}\n  R² Score:
{r2_forest:.2f}\n")

# Store results in a DataFrame for better visualization
results = {
    'Linear Regression': {'MAE': mae_linear, 'R²': r2_linear},
    'Decision Tree Regression': {'MAE': mae_tree, 'R²': r2_tree},
    'Random Forest Regression': {'MAE': mae_forest, 'R²': r2_forest}
}
```

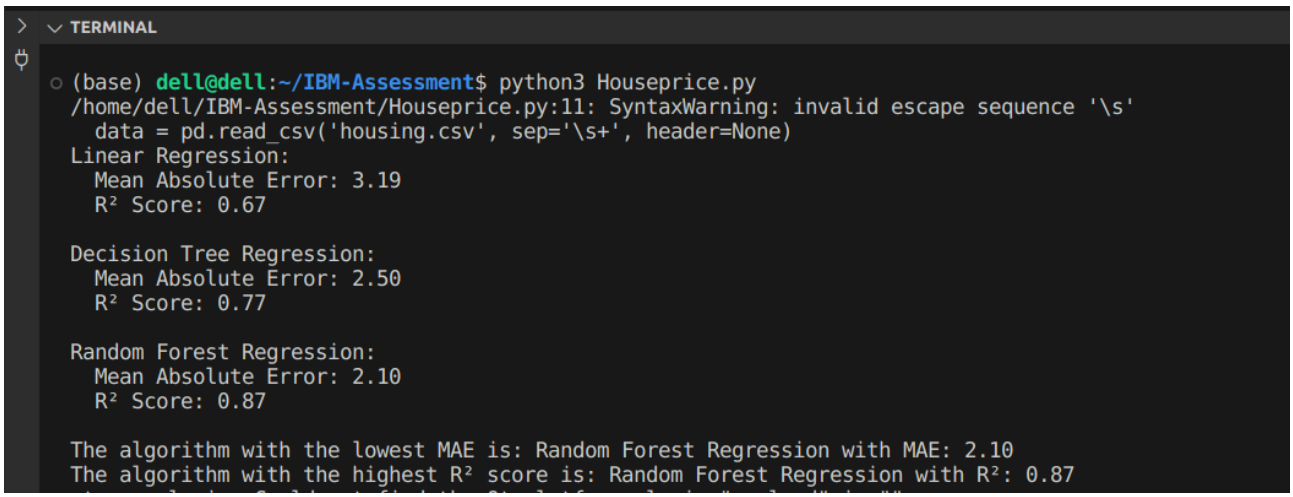
```
results_df = pd.DataFrame(results).T

# Determine the most efficient algorithm based on MAE and R²
best_mae_algorithm = results_df['MAE'].idxmin()
best_r2_algorithm = results_df['R²'].idxmax()

# Print the most efficient algorithm based on both metrics
print(f"The algorithm with the lowest MAE is: {best_mae_algorithm} with MAE:
{results[best_mae_algorithm]['MAE']:.2f}")
print(f"The algorithm with the highest R² score is: {best_r2_algorithm} with R²:
{results[best_r2_algorithm]['R²']:.2f}")

# Plot the results
plt.figure(figsize=(10, 6))
sns.barplot(x=results_df.index, y='MAE', data=results_df)
plt.title('Mean Absolute Error of Different Algorithms')
plt.xlabel('Algorithms')
plt.ylabel('Mean Absolute Error')
plt.xticks(rotation=45)
plt.show()

plt.figure(figsize=(10, 6))
sns.barplot(x=results_df.index, y='R²', data=results_df)
plt.title('R² Score of Different Algorithms')
plt.xlabel('Algorithms')
plt.ylabel('R² Score')
plt.xticks(rotation=45)
plt.show()
```


TERMINAL
(base) dell@dell:~/IBM-Assessment$ python3 Houseprice.py
/home/dell/IBM-Assessment/Houseprice.py:11: SyntaxWarning: invalid escape sequence '\s'
data = pd.read_csv('housing.csv', sep='\s+', header=None)
Linear Regression:
Mean Absolute Error: 3.19
R² Score: 0.67
Decision Tree Regression:
Mean Absolute Error: 2.50
R² Score: 0.77
Random Forest Regression:
Mean Absolute Error: 2.10
R² Score: 0.87
The algorithm with the lowest MAE is: Random Forest Regression with MAE: 2.10
The algorithm with the highest R² score is: Random Forest Regression with R²: 0.87