Step 1: Reading the .csv file of the dataset

In [2]:

```
# Import necessary libraries
import pandas as pd

# Read the .csv file of the dataset
data = pd.read_csv("C:/Users/SMIT YENKAR/OneDrive/Desktop/T.Y .Sub/ML LAB/ho
```

Step 2: Display few observations

In [3]:

```
# Display first few observations of the dataset
print("First few observations of the dataset:")
print(data.head())
```

```
First few observations of the dataset:
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0    -122.23     37.88                41.0        880.0           129.0
1    -122.22     37.86                21.0       7099.0          1106.0
2    -122.24     37.85                52.0       1467.0           190.0
3    -122.25     37.85                52.0       1274.0           235.0
4    -122.25     37.85                52.0       1627.0           280.0

   population  households  median_income  median_house_value ocean_proximity
0       322.0       126.0         8.3252            452600.0        NEAR BAY
1      2401.0      1138.0         8.3014            358500.0        NEAR BAY
2       496.0       177.0         7.2574            352100.0        NEAR BAY
3       558.0       219.0         5.6431            341300.0        NEAR BAY
4       565.0       259.0         3.8462            342200.0        NEAR BAY
```

Step 3: Perform data preprocessing

In [5]:

```
# Handle missing data
data.dropna(inplace=True)  # Drop rows with missing values
```

Step 4: Create the independent and dependent variables

In [8]:

```
# Create the independent and dependent variables
X = data.drop(columns=['median_house_value'])
y = data['median_house_value']
```

Step 5: Standardization of data with one-hot encoding for categorical variables

In [10]:

```
# One-hot encoding for categorical variables
X_encoded = pd.get_dummies(X, columns=['ocean_proximity'])

# Standardization of data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_encoded)
```

Step 6: Split the data into training and test sets

In [11]:

```
# Split the data into training and test sets
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0
```

Step 7: Training Decision tree classifier and RF classifier

In [12]:

```
# Training Decision tree classifier and RF classifier
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor

dt_regressor = DecisionTreeRegressor(random_state=42)
rf_regressor = RandomForestRegressor(random_state=42)

dt_regressor.fit(X_train, y_train)
rf_regressor.fit(X_train, y_train)
```

Out[12]:

```
RandomForestRegressor(random_state=42)
```

Step 8: Apply bagging and boosting algorithm

In [13]:

```
# Apply bagging and boosting algorithm
from sklearn.ensemble import BaggingRegressor, AdaBoostRegressor

bagging_regressor = BaggingRegressor(base_estimator=dt_regressor, n_estimato
boosting_regressor = AdaBoostRegressor(base_estimator=dt_regressor, n_estima

bagging_regressor.fit(X_train, y_train)
boosting_regressor.fit(X_train, y_train)
```

Out[13]:

```
AdaBoostRegressor(base_estimator=DecisionTreeRegressor(random_state=42),
                  n_estimators=10, random_state=42)
```

Step 9: Fit the data in the model to train it

In [14]:

```
# Fit the data in model to train it
bagging_regressor.fit(X_train, y_train)
boosting_regressor.fit(X_train, y_train)
```

Out[14]:

```
AdaBoostRegressor(base_estimator=DecisionTreeRegressor(random_state=42),
                  n_estimators=10, random_state=42)
```

Step 10: Create and fit Stacking Regressor (for regression tasks)

In [17]:

```
# Step 10: Create and fit Stacking Regressor (for regression tasks)
# Create and fit Stacking Regressor
from sklearn.linear_model import LinearRegression
from mlxtend.regressor import StackingRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, BaggingRegressor, AdaBoo

# Define individual regressors
dt_regressor = DecisionTreeRegressor(random_state=42)
rf_regressor = RandomForestRegressor(random_state=42)
bagging_regressor = BaggingRegressor(base_estimator=dt_regressor, n_estimato
boosting_regressor = AdaBoostRegressor(base_estimator=dt_regressor, n_estima

# Fit individual regressors
dt_regressor.fit(X_train, y_train)
rf_regressor.fit(X_train, y_train)
bagging_regressor.fit(X_train, y_train)
boosting_regressor.fit(X_train, y_train)

# Create and fit Stacking Regressor
stacking_regressor = StackingRegressor(regressors=[dt_regressor, rf_regresso
                                       meta_regressor=LinearRegression())

stacking_regressor.fit(X_train, y_train)
```

Out[17]:
```
StackingRegressor(meta_regressor=LinearRegression(),
                  regressors=[DecisionTreeRegressor(random_state=42),
                              RandomForestRegressor(random_state=42),
                              BaggingRegressor(base_estimator=DecisionTreeR
egressor(random_state=42),
                                               random_state=42),
                              AdaBoostRegressor(base_estimator=DecisionTree
Regressor(random_state=42),
                                                n_estimators=10,
                                                random_state=42)])
```

Step 11: Evaluate the performance of each regressor

In [18]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_scor

# Make predictions on the test set
y_pred_dt = dt_regressor.predict(X_test)
y_pred_rf = rf_regressor.predict(X_test)
y_pred_bagging = bagging_regressor.predict(X_test)
y_pred_boosting = boosting_regressor.predict(X_test)
y_pred_stacking = stacking_regressor.predict(X_test)

# Calculate evaluation metrics
mse_dt = mean_squared_error(y_test, y_pred_dt)
mae_dt = mean_absolute_error(y_test, y_pred_dt)
r2_dt = r2_score(y_test, y_pred_dt)

mse_rf = mean_squared_error(y_test, y_pred_rf)
mae_rf = mean_absolute_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)

mse_bagging = mean_squared_error(y_test, y_pred_bagging)
mae_bagging = mean_absolute_error(y_test, y_pred_bagging)
r2_bagging = r2_score(y_test, y_pred_bagging)

mse_boosting = mean_squared_error(y_test, y_pred_boosting)
mae_boosting = mean_absolute_error(y_test, y_pred_boosting)
r2_boosting = r2_score(y_test, y_pred_boosting)

mse_stacking = mean_squared_error(y_test, y_pred_stacking)
mae_stacking = mean_absolute_error(y_test, y_pred_stacking)
r2_stacking = r2_score(y_test, y_pred_stacking)

# Print evaluation metrics
print("Evaluation Metrics:")
print("Decision Tree:")
print("Mean Squared Error:", mse_dt)
print("Mean Absolute Error:", mae_dt)
print("R-squared Score:", r2_dt)
print()

print("Random Forest:")
print("Mean Squared Error:", mse_rf)
print("Mean Absolute Error:", mae_rf)
print("R-squared Score:", r2_rf)
print()

print("Bagging:")
print("Mean Squared Error:", mse_bagging)
print("Mean Absolute Error:", mae_bagging)
print("R-squared Score:", r2_bagging)
print()

print("Boosting:")
print("Mean Squared Error:", mse_boosting)
print("Mean Absolute Error:", mae_boosting)
print("R-squared Score:", r2_boosting)
print()

print("Stacking:")
print("Mean Squared Error:", mse_stacking)
print("Mean Absolute Error:", mae_stacking)
print("R-squared Score:", r2_stacking)
print()
```

```
Evaluation Metrics:
Decision Tree:
Mean Squared Error: 4796653353.900906
Mean Absolute Error: 44264.05505260582
R-squared Score: 0.6456786689990528

Random Forest:
Mean Squared Error: 2383762442.3119
Mean Absolute Error: 31944.929811597747
R-squared Score: 0.8239151718847619

Bagging:
Mean Squared Error: 2641627522.8680353
Mean Absolute Error: 33536.623489111815
R-squared Score: 0.8048670790124656

Boosting:
Mean Squared Error: 2813406794.4551015
Mean Absolute Error: 33609.57621727428
R-squared Score: 0.7921780111027314

Stacking:
Mean Squared Error: 4796653353.900913
Mean Absolute Error: 44264.055052605865
R-squared Score: 0.6456786689990521
```

In [ ]: