In [ ]:

```
Data Preprocessing
```

In [5]:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

#load the dataset
data = fetch_california_housing()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = data.target   #adding target column
```

In [7]:

```python
#check for missing values
print(df.isnull().sum())    #no missing values in the dataset
```

```
MedInc        0
HouseAge      0
AveRooms      0
AveBedrms     0
Population    0
AveOccup      0
Latitude      0
Longitude     0
Target        0
dtype: int64
```

In [9]:

```python
# Splitting data into train and test sets
X = df.drop(columns=['Target'])
y = df['Target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

#Feature Scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [ ]:

```
Linear Regression
```

In [26]:

```python
# Initialize and train the model
lr = LinearRegression()
lr.fit(X_train_scaled, y_train)

# Make predictions
y_pred_lr = lr.predict(X_test_scaled)

# Evaluate
mse_lr = mean_squared_error(y_test, y_pred_lr)
mae_lr = mean_absolute_error(y_test, y_pred_lr)
r2_lr = r2_score(y_test, y_pred_lr)
```

In [ ]:

```
DecisionTree Regressor
```

In [28]:

```
# Initialize and train Decision Tree model
dt = DecisionTreeRegressor(random_state=42)
dt.fit(X_train_scaled, y_train)

# Make predictions
y_pred_dt = dt.predict(X_test_scaled)

# Evaluate
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)
```

In [ ]:

```
Random Forest
```

In [30]:

```
# Initialize and train Random Forest model
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train_scaled, y_train)

# Make predictions
y_pred_rf = rf.predict(X_test_scaled)

# Evaluate
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)
```

In [ ]:

```
Gradient Boosting Regressor
```

In [34]:

```
from sklearn.ensemble import GradientBoostingRegressor
# Initialize and train Gradient Boosting model
gb = GradientBoostingRegressor(n_estimators=100, random_state=42)
gb.fit(X_train_scaled, y_train)

# Make predictions
y_pred_gb = gb.predict(X_test_scaled)

# Evaluate the model
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

mse_gb = mean_squared_error(y_test, y_pred_gb)
mae_gb = mean_absolute_error(y_test, y_pred_gb)
r2_gb = r2_score(y_test, y_pred_gb)

print("Gradient Boosting - MSE:", mse_gb)
print("Gradient Boosting - MAE:", mae_gb)
print("Gradient Boosting - R² Score:", r2_gb)
```

```
Gradient Boosting - MSE: 0.29399901242474274
Gradient Boosting - MAE: 0.37165044848436773
Gradient Boosting - R² Score: 0.7756433164710084
```

In [ ]:

```
SVR
```

In [36]:

```
from sklearn.svm import SVR

svr = SVR(kernel='rbf')
svr.fit(X_train_scaled, y_train)
y_pred_svr = svr.predict(X_test_scaled)

# Evaluation
mse_svr = mean_squared_error(y_test, y_pred_svr)
mae_svr = mean_absolute_error(y_test, y_pred_svr)
r2_svr = r2_score(y_test, y_pred_svr)
```

In [ ]:

```
Comparison
```

In [38]:

```
results = pd.DataFrame({
    "Model": ["Linear Regression", "Decision Tree", "Random Forest", "Gradient Boos
    "MSE": [mse_lr, mse_dt, mse_rf, mse_gb, mse_svr],
    "MAE": [mae_lr, mae_dt, mae_rf, mae_gb, mae_svr],
    "R2 Score": [r2_lr, r2_dt, r2_rf, r2_gb, r2_svr]
})

print(results.sort_values(by="MSE"))
```

```
               Model       MSE       MAE  R2 Score
2      Random Forest  0.255170  0.327425  0.805275
3  Gradient Boosting  0.293999  0.371650  0.775643
4                SVR  0.357004  0.398599  0.727563
1      Decision Tree  0.493969  0.453904  0.623042
0  Linear Regression  0.555892  0.533200  0.575788
```

In [ ]:

```
#based on the results, the best performing model is the "Random Forest". it has the
```

In [ ]:

```
# The worst performing model is the linear regression because it has the highest MS
```

In [ ]:

```

```