In [1]:
```
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
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegress
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_sc
```

Load California Housing Dataset

In [2]:
```
data = fetch_california_housing()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = data.target
```

Display basic info

In [3]:
```
print(df.info())
print(df.describe())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   MedInc      20640 non-null  float64
 1   HouseAge    20640 non-null  float64
 2   AveRooms    20640 non-null  float64
 3   AveBedrms   20640 non-null  float64
 4   Population  20640 non-null  float64
 5   AveOccup    20640 non-null  float64
 6   Latitude    20640 non-null  float64
 7   Longitude   20640 non-null  float64
 8   Target      20640 non-null  float64
dtypes: float64(9)
memory usage: 1.4 MB
None
             MedInc      HouseAge      AveRooms     AveBedrms    Populati
on  \
count  20640.000000  20640.000000  20640.000000  20640.000000  20640.0000
00
mean       3.870671     28.639486      5.429000      1.096675   1425.4767
44
std        1.899822     12.585558      2.474173      0.473911   1132.4621
22
min        0.499900      1.000000      0.846154      0.333333      3.0000
00
25%        2.563400     18.000000      4.440716      1.006079    787.0000
00
50%        3.534800     29.000000      5.229129      1.048780   1166.0000
00
75%        4.743250     37.000000      6.052381      1.099526   1725.0000
00
max       15.000100     52.000000    141.909091     34.066667  35682.0000
00

           AveOccup      Latitude     Longitude        Target
count  20640.000000  20640.000000  20640.000000  20640.000000
mean       3.070655     35.631861   -119.569704      2.068558
std       10.386050      2.135952      2.003532      1.153956
min        0.692308     32.540000   -124.350000      0.149990
25%        2.429741     33.930000   -121.800000      1.196000
50%        2.818116     34.260000   -118.490000      1.797000
75%        3.282261     37.710000   -118.010000      2.647250
max     1243.333333     41.950000   -114.310000      5.000010
```

Check for missing values

In [4]:
```
print("\nMissing values in dataset:")
print(df.isnull().sum())
```

```
Missing values in dataset:
MedInc        0
HouseAge      0
AveRooms      0
AveBedrms     0
Population    0
AveOccup      0
Latitude      0
Longitude     0
Target        0
dtype: int64
```

Splitting Data

In [5]:
```
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, ra
```

Feature Scaling

In [6]:
```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Initialize models

In [7]:
```
models = {
    "Linear Regression": LinearRegression(),
    "Decision Tree": DecisionTreeRegressor(random_state=42),
    "Random Forest": RandomForestRegressor(n_estimators=100, random_state=4
    "Gradient Boosting": GradientBoostingRegressor(n_estimators=100, randor
    "SVR": SVR()
}
```

Model Training and Evaluation

In [8]:
```
results = {}
for name, model in models.items():
    model.fit(X_train_scaled, y_train)
    y_pred = model.predict(X_test_scaled)

    mse = mean_squared_error(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    results[name] = {"MSE": mse, "MAE": mae, "R2 Score": r2}

    print(f"{name} Results:")
    print(f"MSE: {mse:.4f}, MAE: {mae:.4f}, R2 Score: {r2:.4f}\n")
```

```
Linear Regression Results:
MSE: 0.5559, MAE: 0.5332, R2 Score: 0.5758

Decision Tree Results:
MSE: 0.4940, MAE: 0.4539, R2 Score: 0.6230

Random Forest Results:
MSE: 0.2552, MAE: 0.3274, R2 Score: 0.8053

Gradient Boosting Results:
MSE: 0.2940, MAE: 0.3717, R2 Score: 0.7756

SVR Results:
MSE: 0.3570, MAE: 0.3986, R2 Score: 0.7276
```

Convert results to DataFrame and display

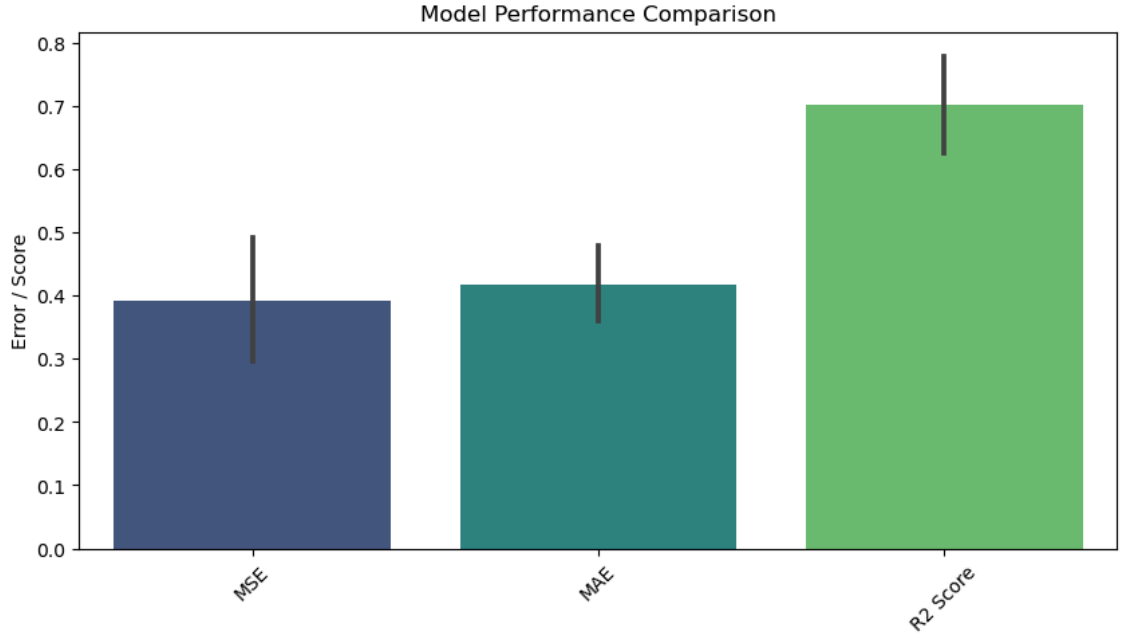
In [10]:
```
results_df = pd.DataFrame(results).T
print("Model Performance Comparison:")
print(results_df)
```

```
Model Performance Comparison:
                           MSE       MAE  R2 Score
Linear Regression  0.555892  0.533200  0.575788
Decision Tree      0.493969  0.453904  0.623042
Random Forest      0.255170  0.327425  0.805275
Gradient Boosting  0.293999  0.371650  0.775643
SVR                0.357004  0.398599  0.727563
```

Visualization

plt.figure(figsize=(10, 5)) sns.barplot(data=results_df, palette='viridis') plt.title("Model Performance Comparison") plt.ylabel("Error / Score") plt.xticks(rotation=45) plt.show()

In [12]:


Model Performance Comparison
Error / Score
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
MSE
MAE
R2 Score

In [ ]: