```
import pandas as pd
import numpy as np
df=pd.read_csv('/content/boston.csv')
df.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2

Next steps: Generate code with df New interactive sheet

```
df.columns
```

```
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT', 'MEDV'],
      dtype='object')
```

```
x=df.drop('MEDV', axis=1)
y=df['MEDV']
```

```
x=np.array(x)
y=np.array(y).reshape(-1, 1)
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test=train_test_split(
    x,y, test_size=0.3, random_state=42
)
```

```
from sklearn.neighbors import KNeighborsRegressor
knn=KNeighborsRegressor()
```

```
from sklearn.metrics import mean_squared_error,r2_score
```

```
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error

print("GridSearchCV and mean_squared_error imported successfully.")
```

```
GridSearchCV and mean_squared_error imported successfully.
```

```
param_grid = {
    'n_neighbors': list(range(1, 11)),
    'weights': ['uniform', 'distance']
}

print("Hyperparameter grid defined successfully.")
print(param_grid)
```

```
Hyperparameter grid defined successfully.
{'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], 'weights': ['uniform', 'distance']}
```

```
grid_search = GridSearchCV(knn, param_grid, cv=5, scoring='neg_mean_squared_error')
grid_search.fit(x_train, y_train)

print("GridSearchCV completed successfully.")
```

```
GridSearchCV completed successfully.
```

```
print(f"Best parameters: {grid_search.best_params_}")
print(f"Best score (negative mean squared error): {grid_search.best_score_}")
```

```
Best parameters: {'n_neighbors': 9, 'weights': 'distance'}
Best score (negative mean squared error): -45.32862141323132
```

```
best_knn = grid_search.best_estimator_
y_pred = best_knn.predict(x_test)
```

```
mse = mean_squared_error(y_test, y_pred)

print(f"Mean Squared Error on test data with best model: {mse}")
```

```
Mean Squared Error on test data with best model: 33.47533286244532
```

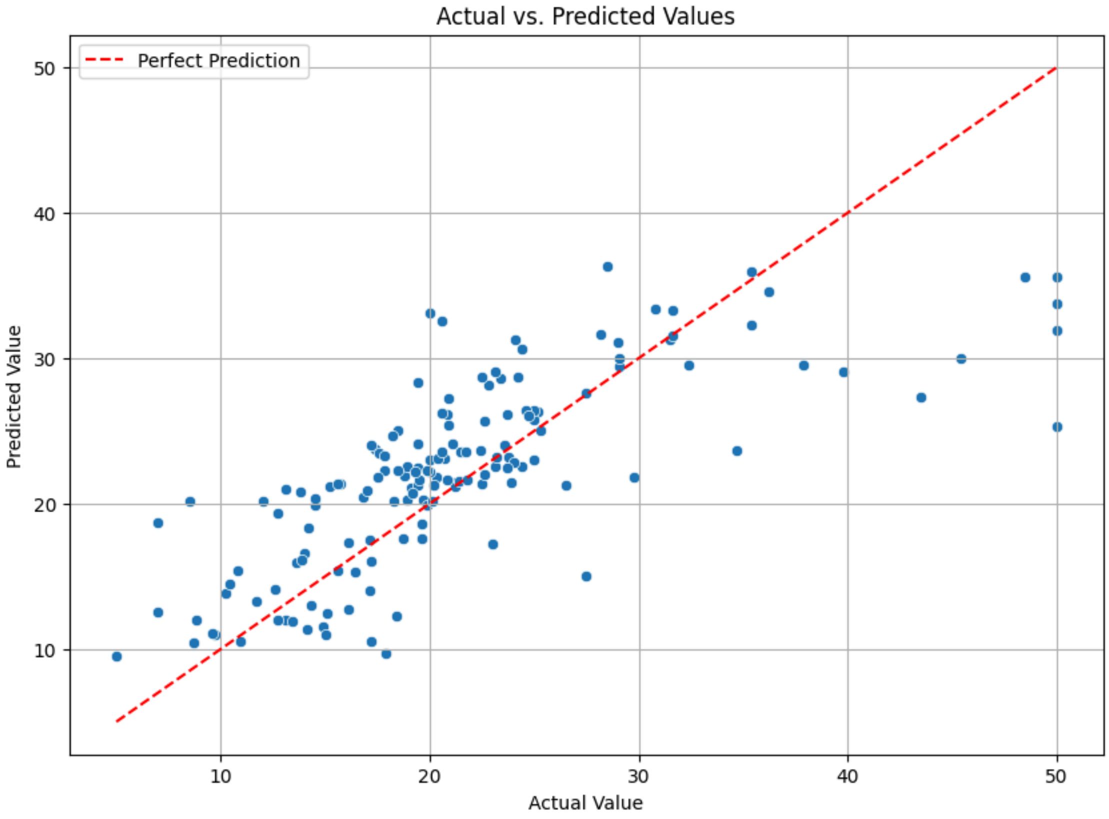
```
rmse = np.sqrt(mse)

print(f"Mean Squared Error (MSE): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
```

```
Mean Squared Error (MSE): 33.47533286244532
Root Mean Squared Error (RMSE): 5.78578714285665
```

```
from sklearn.metrics import r2_score

r_squared = r2_score(y_test, y_pred)

print(f"R-squared: {r_squared}")
```

```
R-squared: 0.5507456426641709
```

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 7))
sns.scatterplot(x='Actual Value', y='Predicted Value', data=results_df)
plt.plot([results_df['Actual Value'].min(), results_df['Actual Value'].max()],
         [results_df['Actual Value'].min(), results_df['Actual Value'].max()],
         color='red', linestyle='--', label='Perfect Prediction')
plt.title('Actual vs. Predicted Values')
plt.xlabel('Actual Value')
plt.ylabel('Predicted Value')
plt.legend()
plt.grid(True)
plt.show()
```



```
best_knn = grid_search.best_estimator_
y_pred = best_knn.predict(x_train)
mse = mean_squared_error(y_train, y_pred)

print(f"Mean Squared Error on test data with best model: {mse}")
```

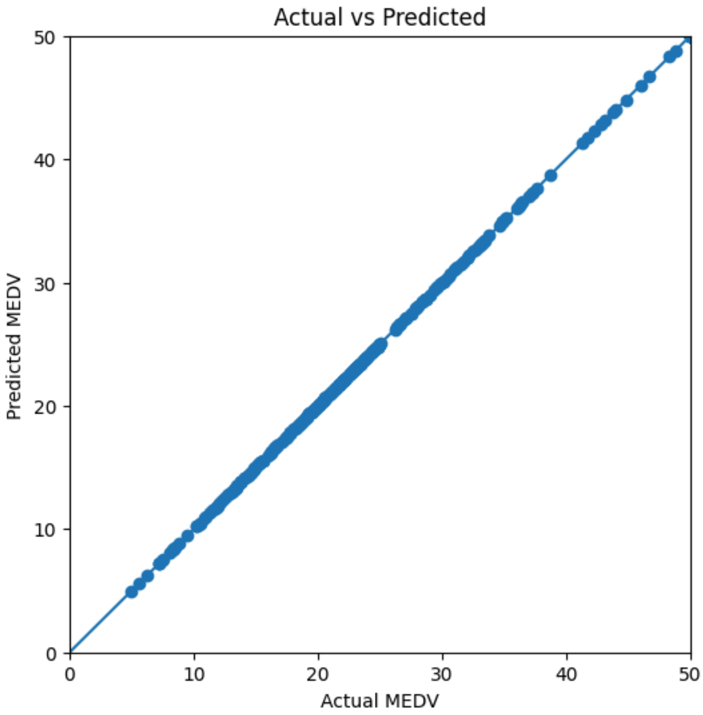
```
Mean Squared Error on test data with best model: 0.0
```

```
from sklearn.metrics import mean_squared_error,r2_score
```

```
y_train_pred = best_knn.predict(x_train)
mse_train=mean_squared_error(y_train,y_train_pred)
rms_traine=np.sqrt(mse_train)
r2_train=r2_score(y_train,y_train_pred)

print(f"Training MSE: {mse_train}")
print(f"Training RMSE: {rms_traine}")
print(f"Training R-squared: {r2_train}")
```

```
Training MSE: 0.0
Training RMSE: 0.0
Training R-squared: 1.0
```

```
import matplotlib.pyplot as plt
plt.figure(figsize=(6,6))
plt.scatter(y_train,y_pred)
#plot prediction line
max_val=max(y_test.max(),y_train_pred.max())
plt.plot([0,max_val],[0,max_val], )
#start axes from 0
plt.xlim(0,max_val)
plt.ylim(0,max_val)
plt.xlabel('Actual MEDV')
plt.ylabel('Predicted MEDV')
plt.title('Actual vs Predicted')
plt.show()
```


Actual vs Predicted
Predicted MEDV
Actual MEDV