House prediction analysis

Predicting house prices ties directly to real estate, economics, and business decisions.

The features listed are standard variables from this dataset:

- CRIM: Per capita crime rate by town.
- ZN: Proportion of residential land zoned for lots over 25,000 sq.ft.
- INDUS: Proportion of non-retail business acres per town.
- CHAS: Charles River dummy variable (1 if tract bounds river; 0 otherwise).
- NOX: Nitric oxides concentration (parts per 10 million).
- RM: Average number of rooms per dwelling.
- AGE: Proportion of owner-occupied units built prior to 1940.
- DIS: Weighted distances to five Boston employment centers.
- RAD: Index of accessibility to radial highways.
- TAX: Full-value property-tax rate per $10,000.
- PTRATIO: Pupil-teacher ratio by town.
- B: 1000(Bk - 0.63)^2 where Bk is the proportion of Black people by town.
- LSTAT: % lower status of the population.
- MEDV: Median value of owner-occupied homes in $1000s. (This is the target variable)

Import libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
import warnings
warnings.filterwarnings('ignore')
import os
```

Loading the dataset

In [2]:
```
# Define column names based on Boston Housing dataset
columns = [
    'CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE',
    'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT', 'MEDV'
]

# Load data
df = pd.read_csv('/home/user/Documents/House prediction/house Predi
ction Data Set.csv', delim_whitespace=True, names=columns)
```

In [3]:
```
# Display the first 10 rows of the dataframe
df.head(10)
```

Out[3]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90
5	0.02985	0.0	2.18	0	0.458	6.430	58.7	6.0622	3	222.0	18.7	394.12
6	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311.0	15.2	395.60
7	0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311.0	15.2	396.90
8	0.21124	12.5	7.87	0	0.524	5.631	100.0	6.0821	5	311.0	15.2	386.63
9	0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311.0	15.2	386.71

In [4]:
```
# Display the column names of the dataframe
df.columns
```

Out[4]:
```
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'R
AD', 'TAX',
       'PTRATIO', 'B', 'LSTAT', 'MEDV'],
      dtype='object')
```

In [5]:
```
# Display the info of the dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    int64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    int64
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 55.5 KB
```

In [6]:
```
# Display the statistical summary of the dataframe
df.describe()
```

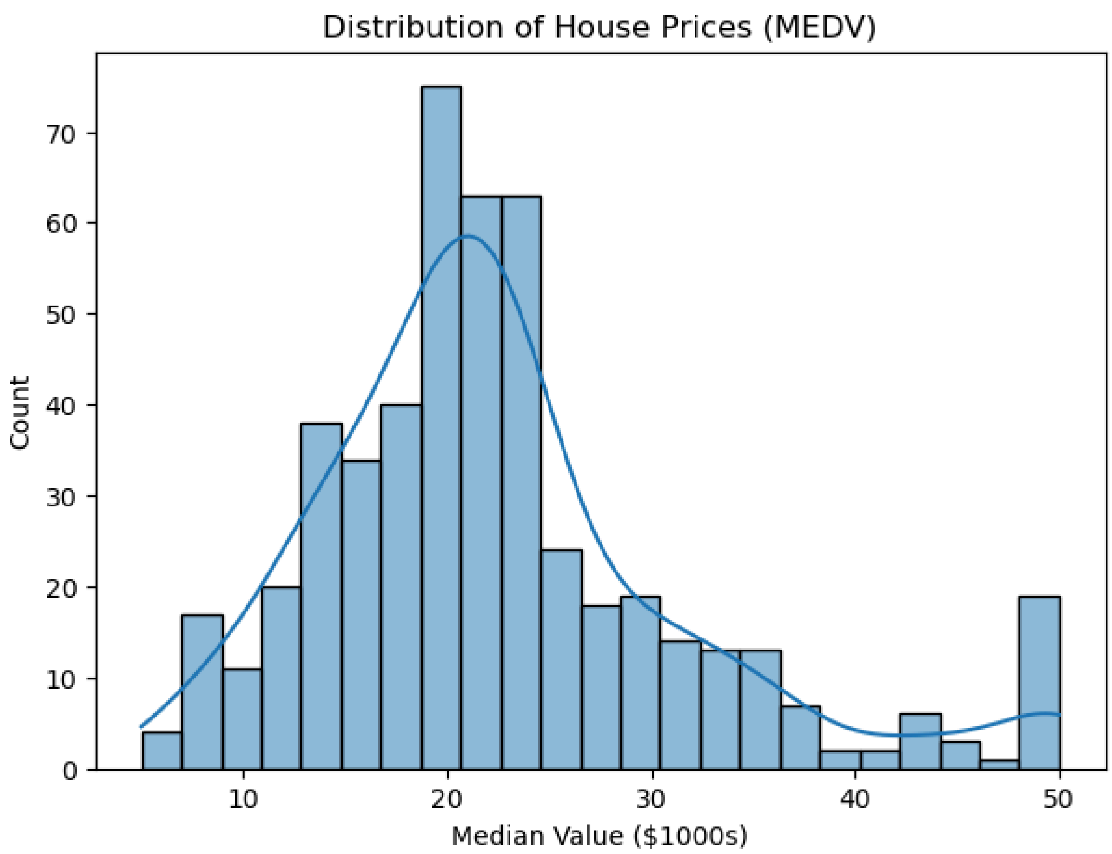
Out[6]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000

In [7]:
```
# Drop duplicate rows from the dataframe
df.drop_duplicates(inplace=True)
```

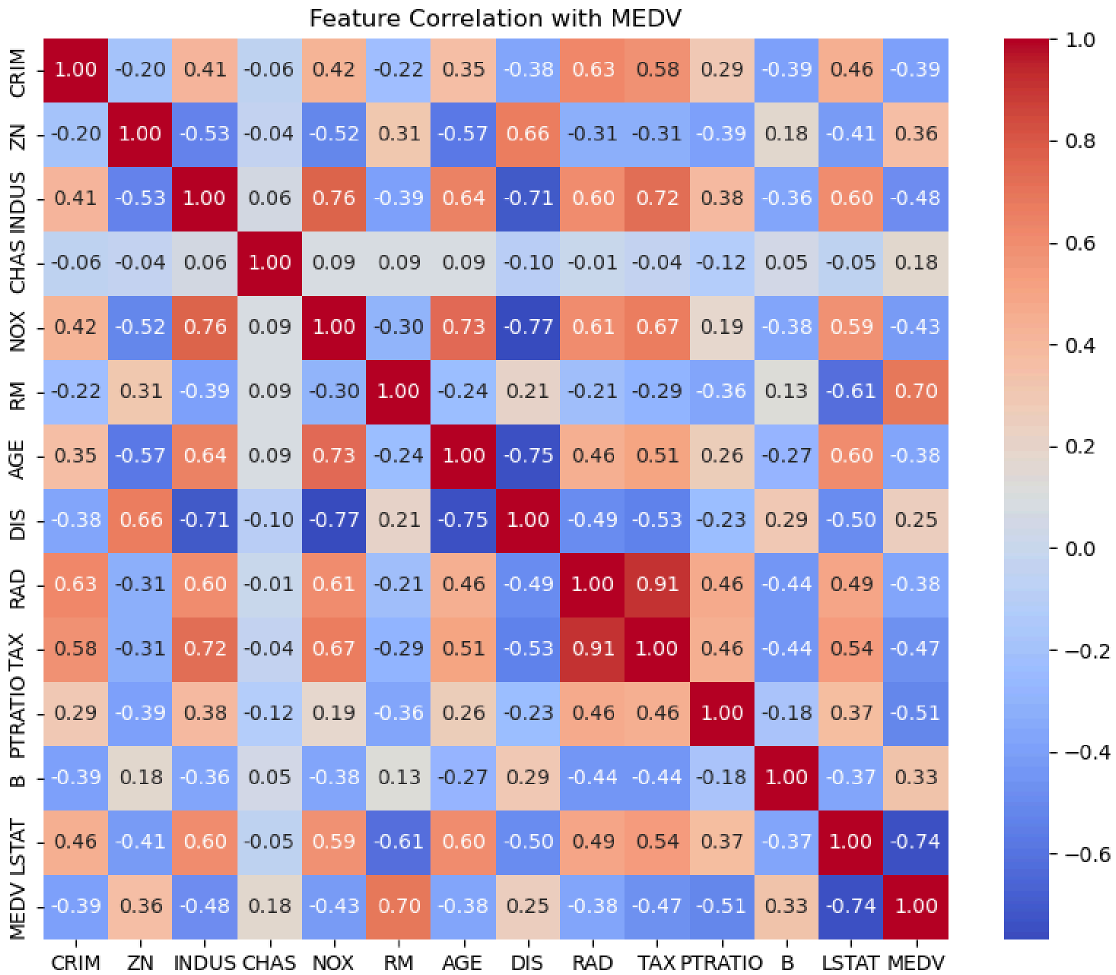
Data exploration

In [8]:
```
# Plot the distribution of the target variable 'MEDV'
plt.figure(figsize=(7,5))
sns.histplot(df['MEDV'], kde=True)
plt.title('Distribution of House Prices (MEDV)')
plt.xlabel('Median Value ($1000s)')
plt.show()
```



In [9]:
```
# the data distribution of the house prices is right-skewed, indicating that
# there are more lower-priced houses than higher-priced ones.
```

In [10]:
```
#correlation heatmap
plt.figure(figsize=(10,8))
corr = df.corr()
sns.heatmap(corr, annot=True, fmt=".2f", cmap='coolwarm')
plt.title('Feature Correlation with MEDV')
plt.show()
```



In [11]:
```
# Strongest Positive Correlation: RM (Average number of rooms) has a correlation of 0.70 with MEDV. This makes intuitive sense:
# houses with more rooms tend to be more valuable. Strongest Negative Correlation: LSTAT (% lower status of the population) has a
# correlation of -0.74 with MEDV. This also makes sense: areas with a higher percentage of lower-status residents tend to have lower
# home values. PTRATIO (Pupil-teacher ratio) is negatively correlated (-0.51), suggesting schools with fewer students per teacher are
# in more expensive areas.DIS (Distance to employment centers) is negatively correlated (-0.43), meaning homes closer to work are
# generally more expensive.RAD (Accessibility to highways) is positively correlated (0.46), indicating better highway access can
# increase value. NOX and INDUS are strongly positively correlated (0.76), which is logical as industrial areas often have higher
# pollution.TAX and PTRATIO are also positively correlated (0.91), suggesting towns with higher taxes might also have higher
# pupil-teacher ratios.
```

In [12]:
```
X = df.drop('MEDV', axis=1)
y = df['MEDV']
```

In [13]:
```
# Train-test split (80-20)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_siz
e=0.2, random_state=42)
```

Linear regression

In [14]:
```
# Initialize and train
lr = LinearRegression()
lr.fit(X_train, y_train)
```

Out[14]:

▾ LinearRegression ⓘ ? (https://scikit-learn.org/1.7/modules/generated/sklearn.linear_model.L

Parameters

In [15]:
```
# Predict
y_pred_lr = lr.predict(X_test)
```

In [16]:
```
# Evaluate
mse_lr = mean_squared_error(y_test, y_pred_lr)
rmse_lr = np.sqrt(mse_lr)
r2_lr = r2_score(y_test, y_pred_lr)
```

In [17]:
```
print(f"Linear Regression Results:")
print(f"R²: {r2_lr:.4f}")
print(f"RMSE: {rmse_lr:.4f}")
```

```
Linear Regression Results:
R²: 0.6688
RMSE: 4.9286
```

In [18]:
```
# The R2 value of 0.6688 indicates that approximately 66.88% of the
variance in house prices can be explained by the features used
# in the model.
# RMSE of 4.9286 means that, on average, the model's predictions ar
e off by about $4,928.60.
```

Random Forest classifier

In [19]:
```
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
```

In [20]:
```
mse_rf = mean_squared_error(y_test, y_pred_rf)
rmse_rf = np.sqrt(mse_rf)
r2_rf = r2_score(y_test, y_pred_rf)
```

In [21]:
```
print(f"\nRandom Forest Results:")
print(f"R²: {r2_rf:.4f}")
print(f"RMSE: {rmse_rf:.4f}")
```

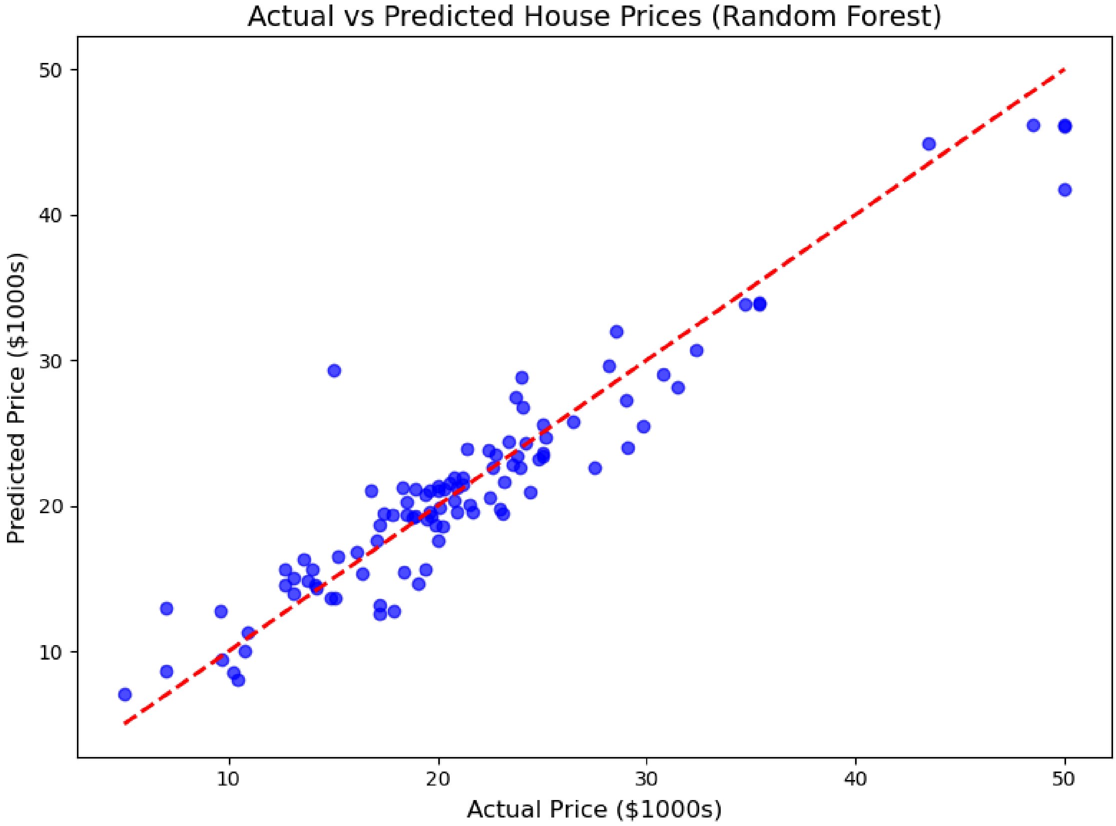
```
Random Forest Results:
R²: 0.8923
RMSE: 2.8110
```

In [22]:
```
# The results from the random forest: R2 of 0.8923 indicate that ab
out 89.23% of the variance in house prices is explained by the mode
l.
# RMSE of 2.8110 indicate that on average the modesl's predictions
are off by about $2,811.00. And the random forest model is outperfo
rming
# the linear regression model significantly in both R2 and RMSE met
rics.
```

Visualization

In [23]:
```
os.makedirs('images', exist_ok=True)
```

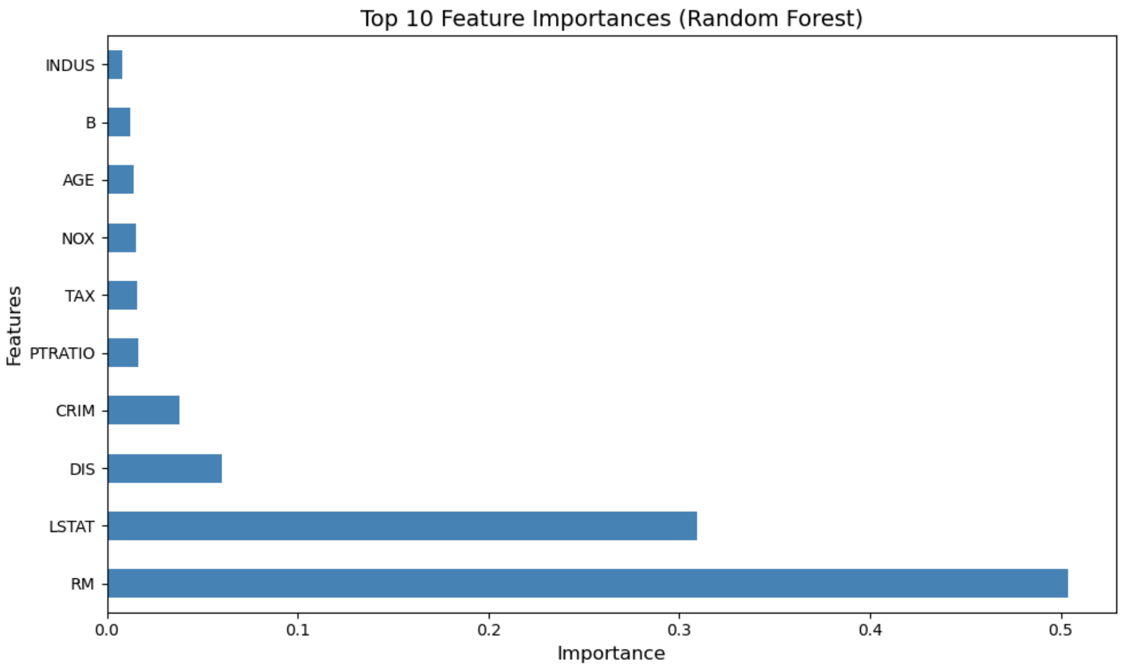
In [30]:
```
# Plot Actual vs Predicted for Linear Regression
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred_rf, alpha=0.7, color='blue')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max
()], 'r--', lw=2)
plt.title('Actual vs Predicted House Prices (Random Forest)', fonts
ize=14)
plt.xlabel('Actual Price ($1000s)', fontsize=12)
plt.ylabel('Predicted Price ($1000s)', fontsize=12)
plt.tight_layout()
plt.savefig('images/actual_vs_predicted.png', dpi=300, bbox_inches
='tight')
plt.show()
```



In [25]:
```
# Most blue dots are clustered around the red dashed line — especia
lly in the middle range (actual prices ~15–35).
# This means the model is doing a decent job predicting average-pri
ced homes.
# For houses with actual prices > 40 (i.e., >$40,000), many predict
ed values fall below the red line.
# The model underestimates expensive homes.
# For very low actual prices (<10), some predictions are higher tha
n they should be (dots above the red line).
# The model overestimates cheap homes.
# Overall, while the model captures general average trends, it stru
ggles with extreme values on both ends.
```

In [31]:
```
# Feature Importance from Random Forest
feat_importances = pd.Series(rf.feature_importances_, index=X.colu
mns)
top_feats = feat_importances.nlargest(10)

plt.figure(figsize=(10, 6))
top_feats.plot(kind='barh', color='steelblue')
plt.title('Top 10 Feature Importances (Random Forest)', fontsize=1
4)
plt.xlabel('Importance', fontsize=12)
plt.ylabel('Features', fontsize=12)
plt.tight_layout()
plt.savefig('images/feature_importance_top10.png', dpi=300, bbox_i
nches='tight')
plt.show()
```



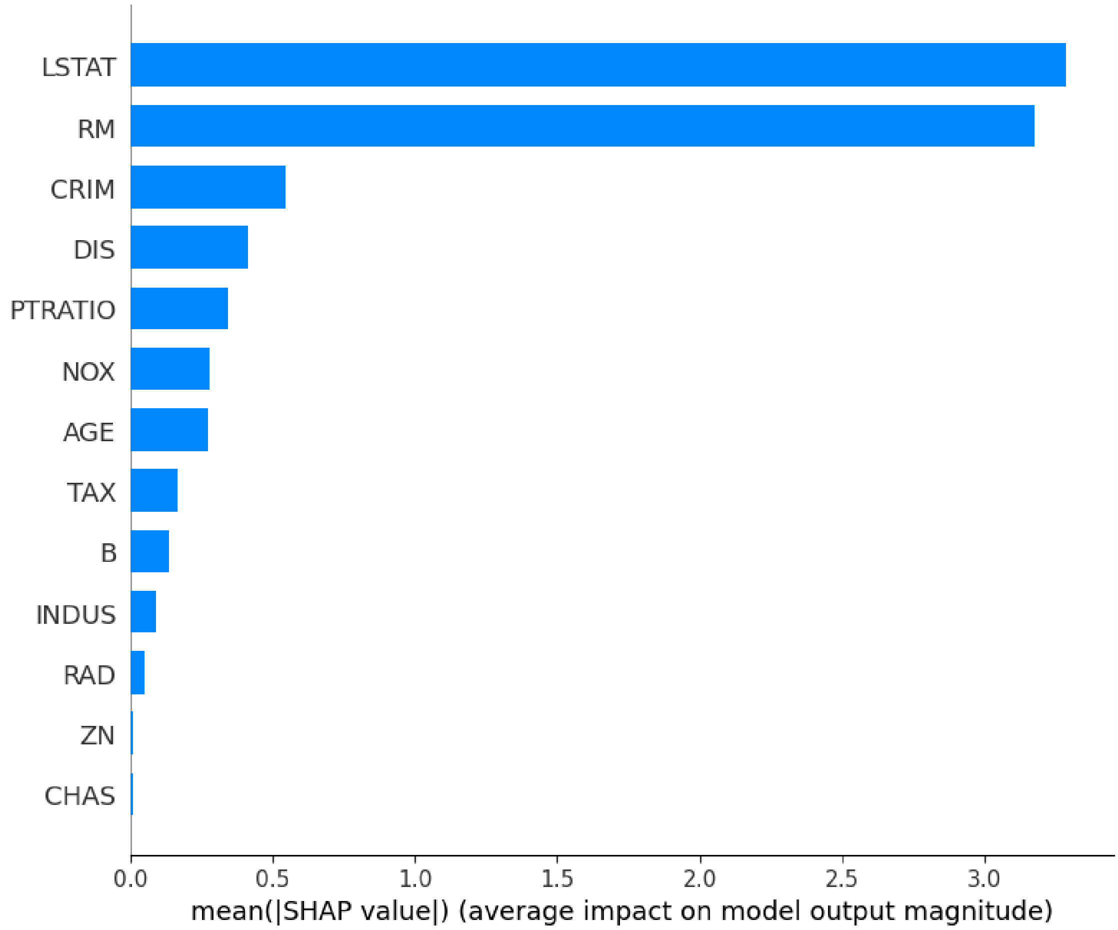
In [27]:
```
# Of all the variables, the number of rooms per dwelling (RM) is th
e single biggest predictor of house price,
# followed by the percentage of lower-status residents (LSTAT). (DI
S) Weighted distance to employment centers is the
# third most important feature. Other features play minor roles.
```

In [28]:
```
#!pip install shap
```

In [29]:
```
# Plot SHAP values for Random Forest
import shap

explainer = shap.TreeExplainer(rf)
shap_values = explainer.shap_values(X_test)

# Summary plot
shap.summary_plot(shap_values, X_test, plot_type="bar")
```



Key Insights from the Plot

Top 2 Most Influential Features

- LSTAT — Highest bar (~3.3+) % lower status of the population. Strong negative effect on house prices. The model uses this heavily to predict lower prices.

- RM — Second highest (~3.2) Average number of rooms per dwelling. Strong positive effect — more rooms → higher predicted price.

Moderately Important Features

- CRIM (crime rate) — ~0.6
- DIS (distance to employment centers) — ~0.5
- PTRATIO (pupil-teacher ratio) — ~0.4 These still have noticeable influence but less than LSTAT and RM.

Low-Impact Features

- NOX, AGE, TAX, B, INDUS — small bars (~0.2–0.3)
- RAD, ZN, CHAS — very tiny bars (<0.1) These contribute minimally to the model's predictions — possibly because they're redundant, noisy, or weakly correlated with MEDV.