Task

give code for this data set

Here is all the data you need: "boston.csv"

Double-click (or enter) to edit

Data loading

Subtask:

Load the Boston housing dataset from the "boston.csv" file.

Reasoning: Load the dataset and display the first few rows and the shape to verify the data is loaded correctly.

```
import pandas as pd

try:
    df = pd.read_csv('boston.csv')
    display(df.head())
    print(df.shape)
except FileNotFoundError:
    print("Error: 'boston.csv' not found. Please ensure the file exists in the current di
except pd.errors.ParserError:
    print("Error: Could not parse the CSV file. Please check the file format.")
except Exception as e:
    print(f"An unexpected error occurred: {e}")
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LS
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	

Data exploration

Subtask:

Explore the loaded Boston Housing dataset to understand its characteristics.

Reasoning: Explore the data by examining the first few rows, data types, descriptive statistics, missing values, and visualize the distributions of numerical features.

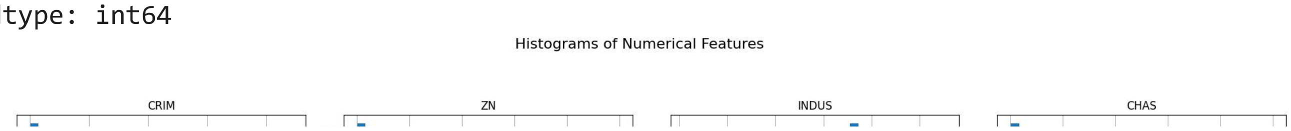
```
# Data Overview
display(df.head())
print(df.dtypes)

# Descriptive Statistics
display(df.describe())

# Missing Values
print(df.isnull().sum())

# Data Distribution (Histograms)
import matplotlib.pyplot as plt
df.hist(bins=30, figsize=(20, 15))
plt.suptitle('Histograms of Numerical Features', fontsize=16)
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()

# Correlation Matrix (Heatmap)
import seaborn as sns
plt.figure(figsize=(12, 10))
correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix of Numerical Features')
plt.show()
```

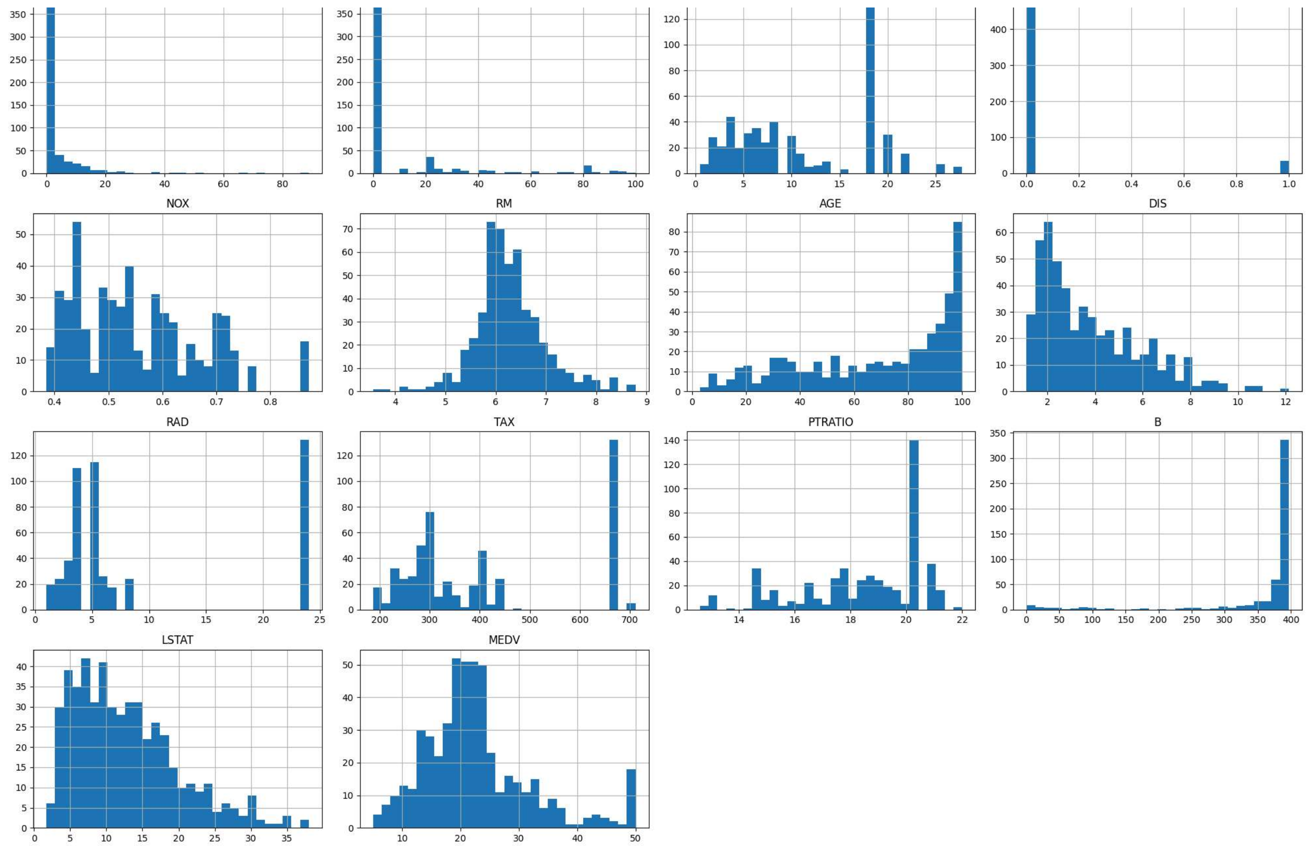
	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LS
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	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	

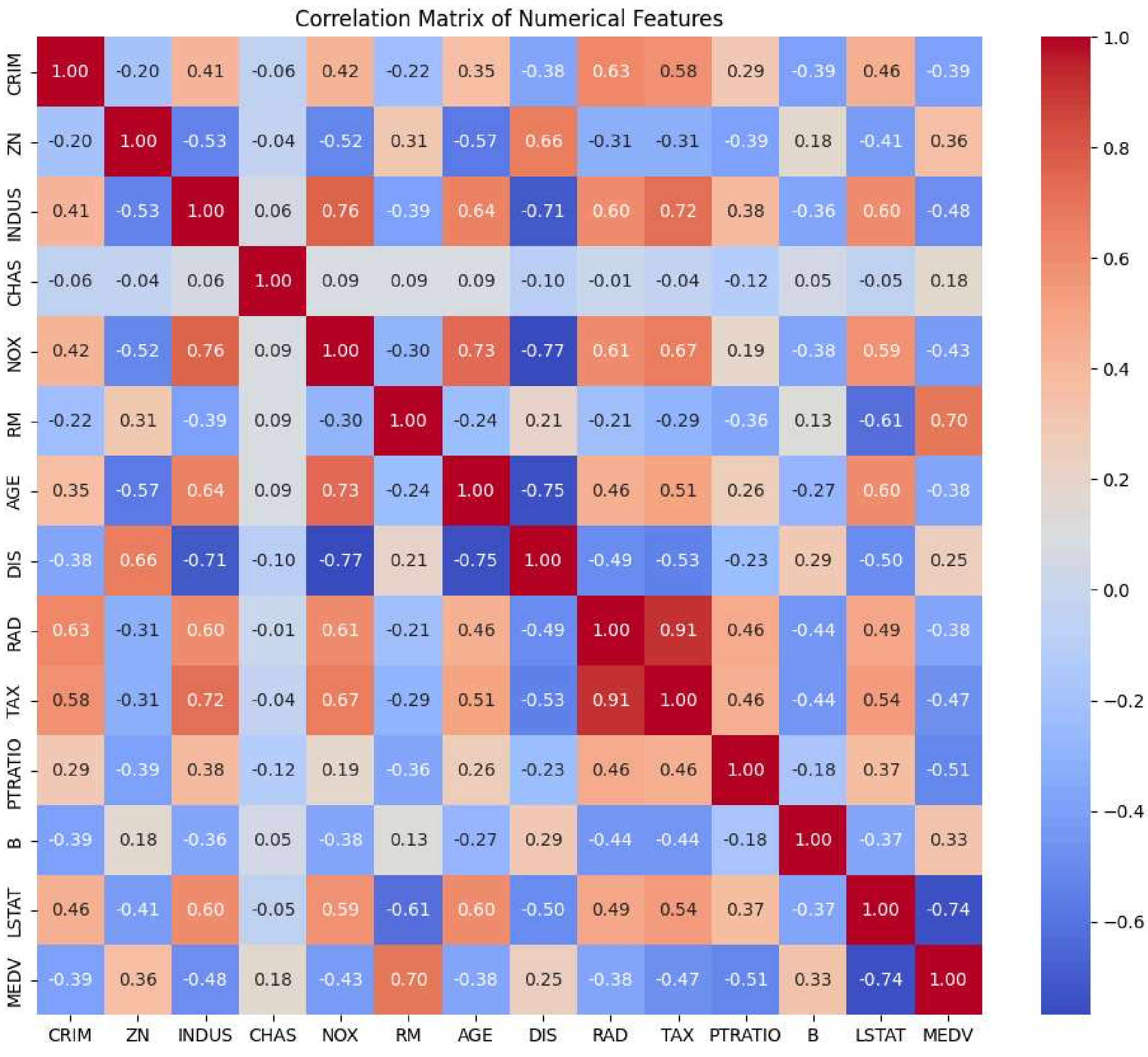
```
CRIM       float64
ZN         float64
INDUS      float64
CHAS         int64
NOX        float64
RM         float64
AGE        float64
DIS        float64
RAD          int64
TAX        float64
PTRATIO    float64
B          float64
LSTAT      float64
MEDV       float64
dtype: object
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AG
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.00000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.5749
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.1488
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.9000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.0250
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.5000
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.0750
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.0000

```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64
```

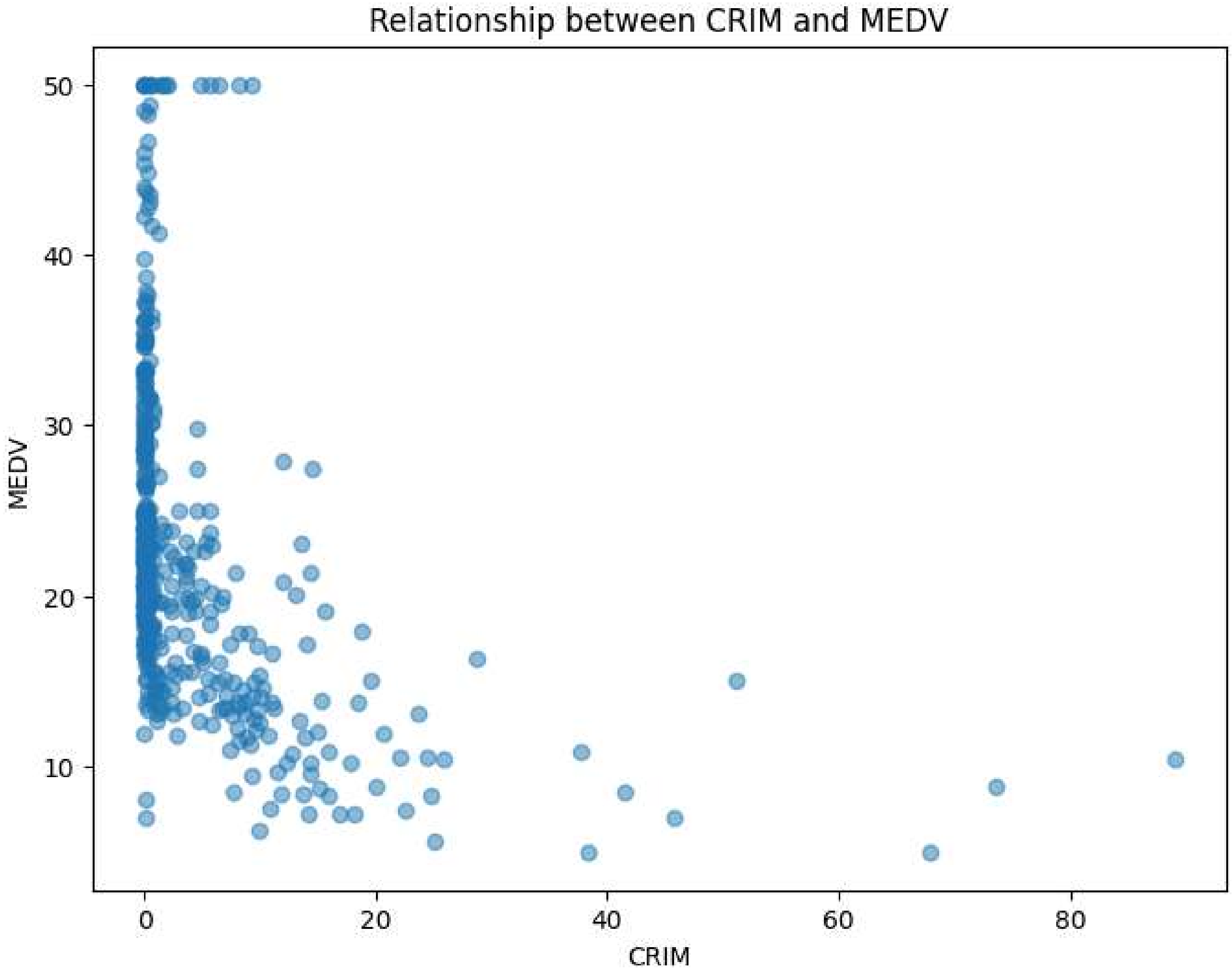


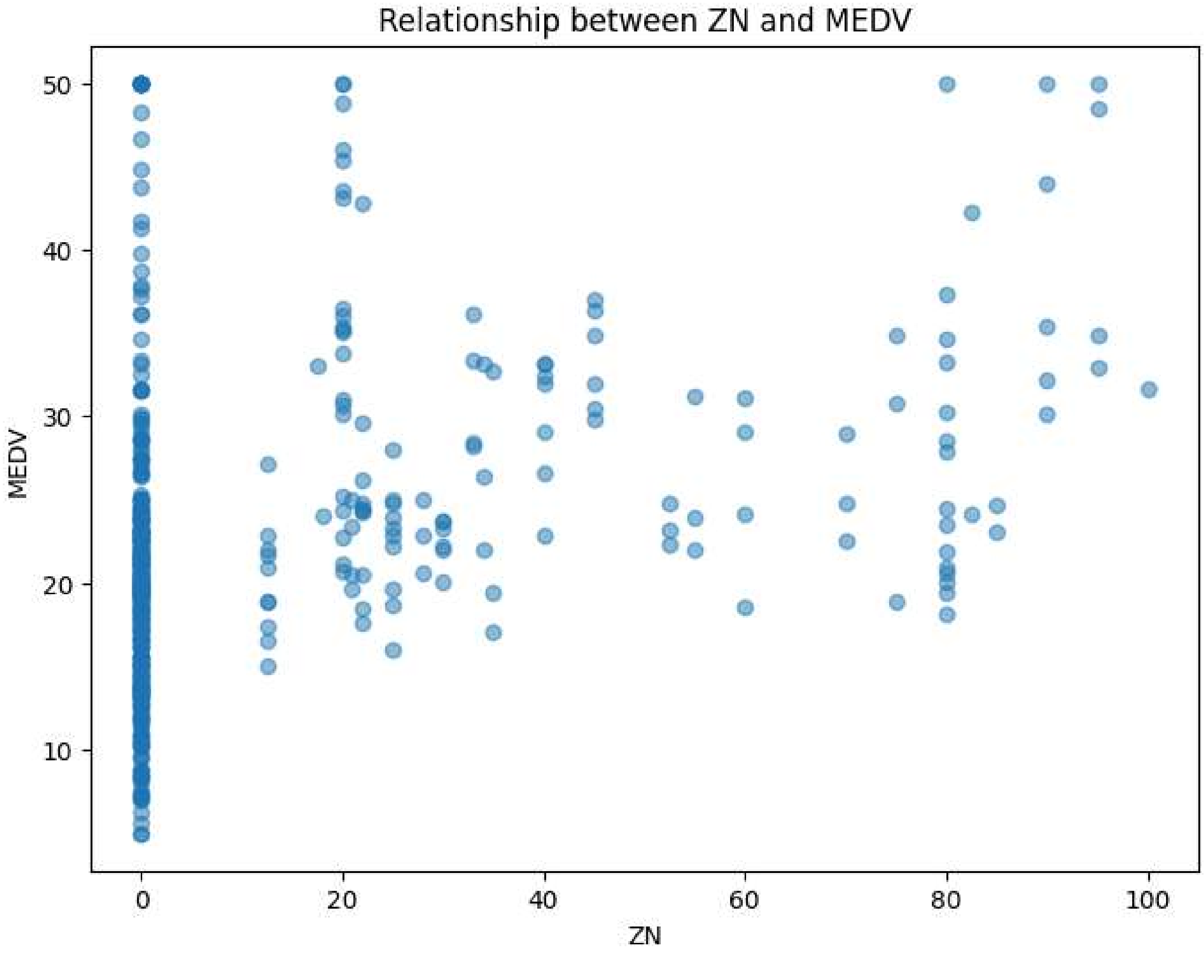

NOX
RM
AGE
DIS
RAD
TAX
PTRATIO
B
LSTAT
MEDV

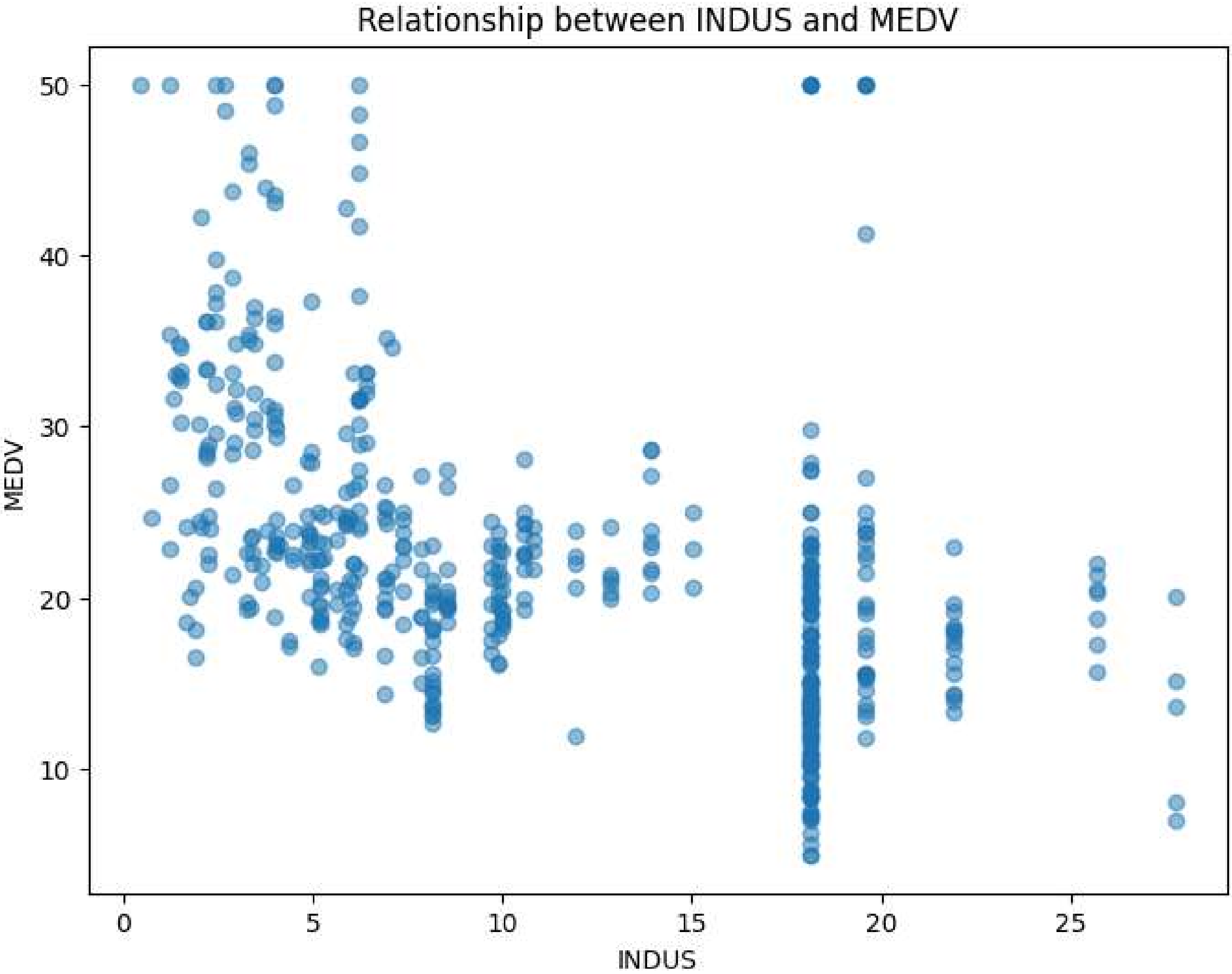

Correlation Matrix of Numerical Features
CRIM
ZN
INDUS
CHAS
NOX
RM
AGE
DIS
RAD
TAX
PTRATIO
B
LSTAT
MEDV

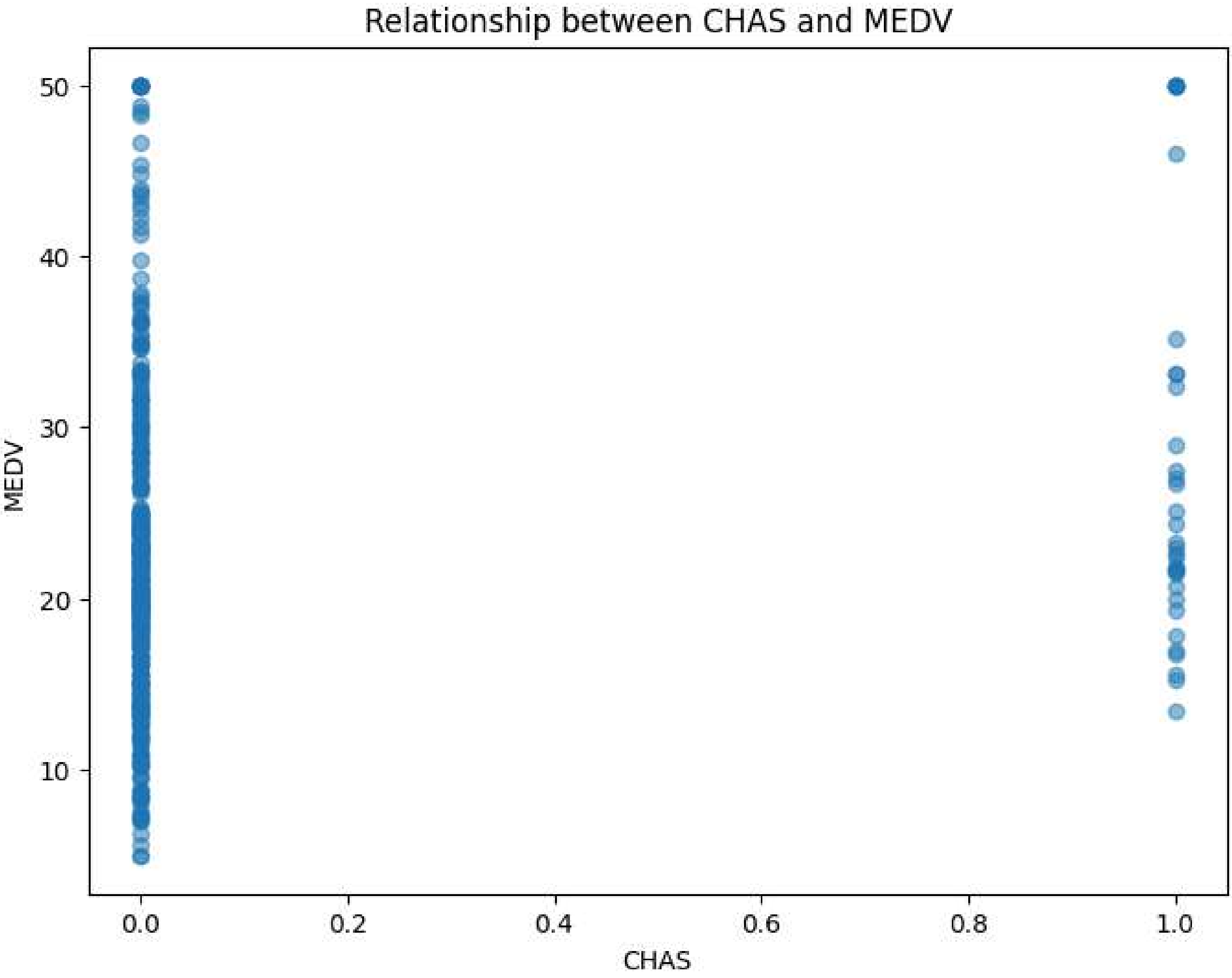
Reasoning: Investigate the relationship between features and the target variable 'MEDV' using scatter plots.

```
import matplotlib.pyplot as plt

# Feature Relationships (Scatter Plots)
for col in df.columns:
    if col != 'MEDV':
        plt.figure(figsize=(8, 6))
        plt.scatter(df[col], df['MEDV'], alpha=0.5)
        plt.title(f'Relationship between {col} and MEDV')
        plt.xlabel(col)
        plt.ylabel('MEDV')
        plt.show()
```

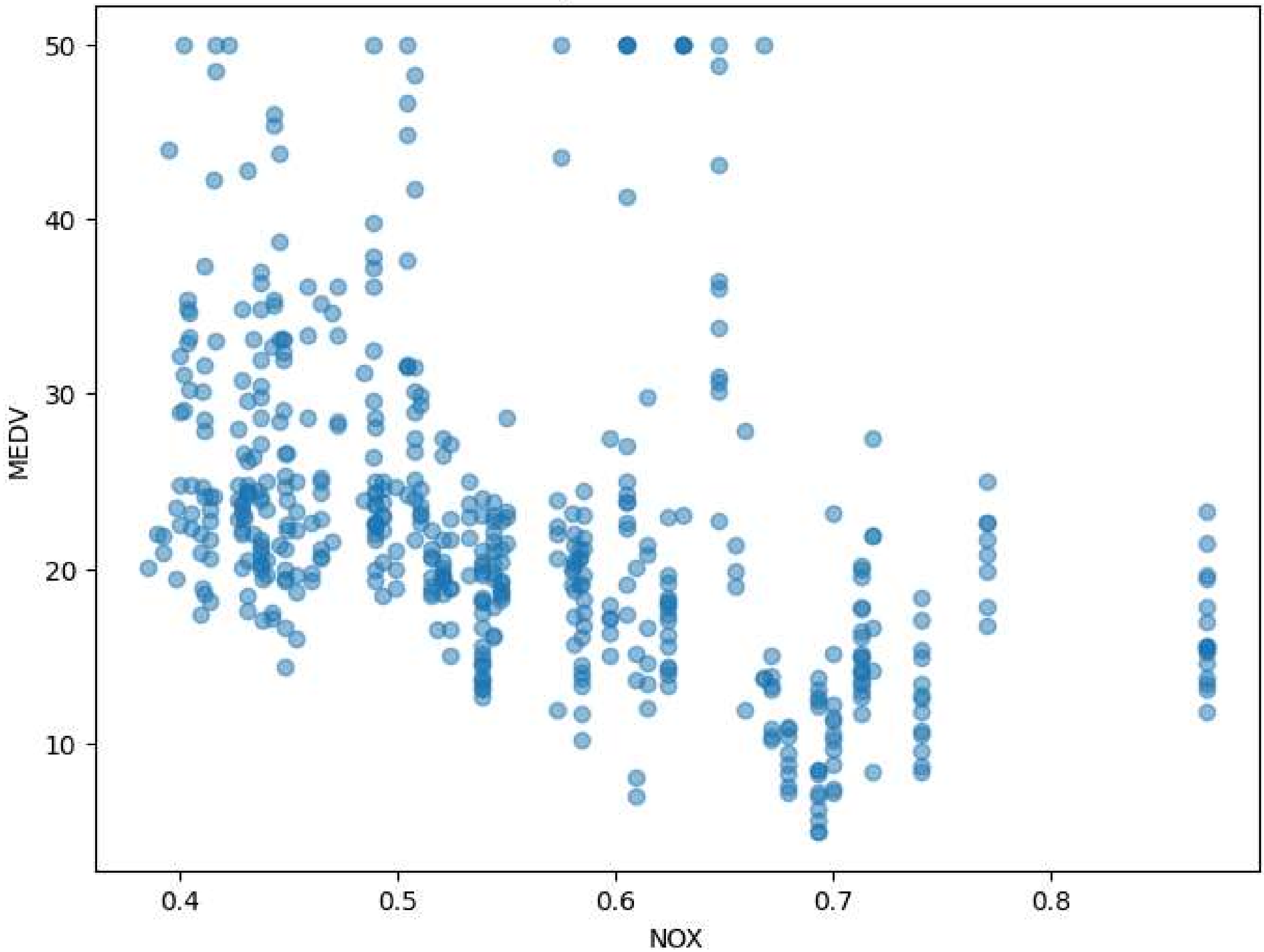

Relationship between CRIM and MEDV
MEDV
CRIM
50
40
30
20
10
0
20
40
60
80


Relationship between ZN and MEDV
MEDV
ZN
50
40
30
20
10
0
20
40
60
80
100

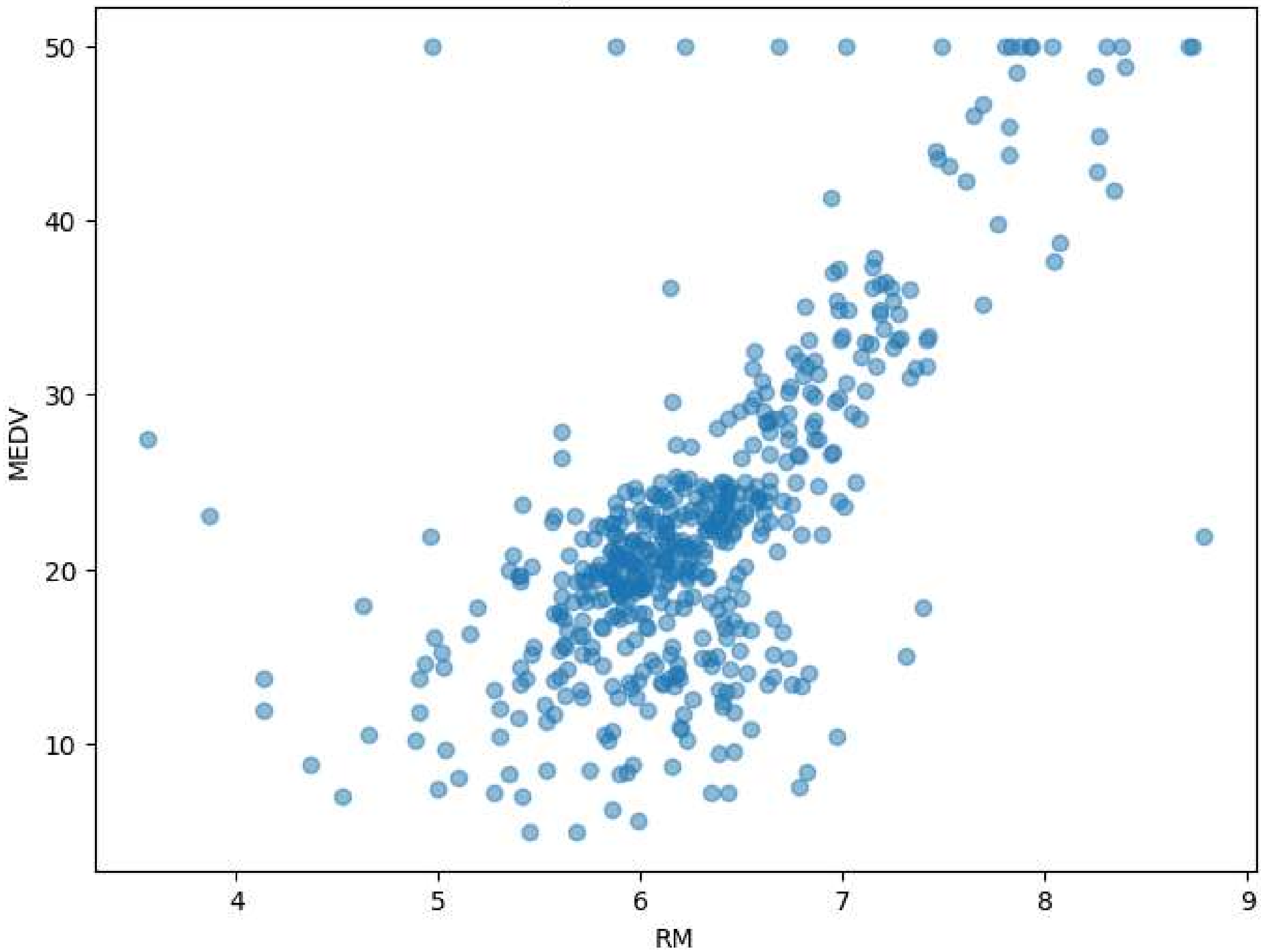

Relationship between INDUS and MEDV
MEDV
INDUS
50
40
30
20
10
0
5
10
15
20
25


Relationship between CHAS and MEDV
MEDV
CHAS
50
40
30
20
10
0.0
0.2
0.4
0.6
0.8
1.0

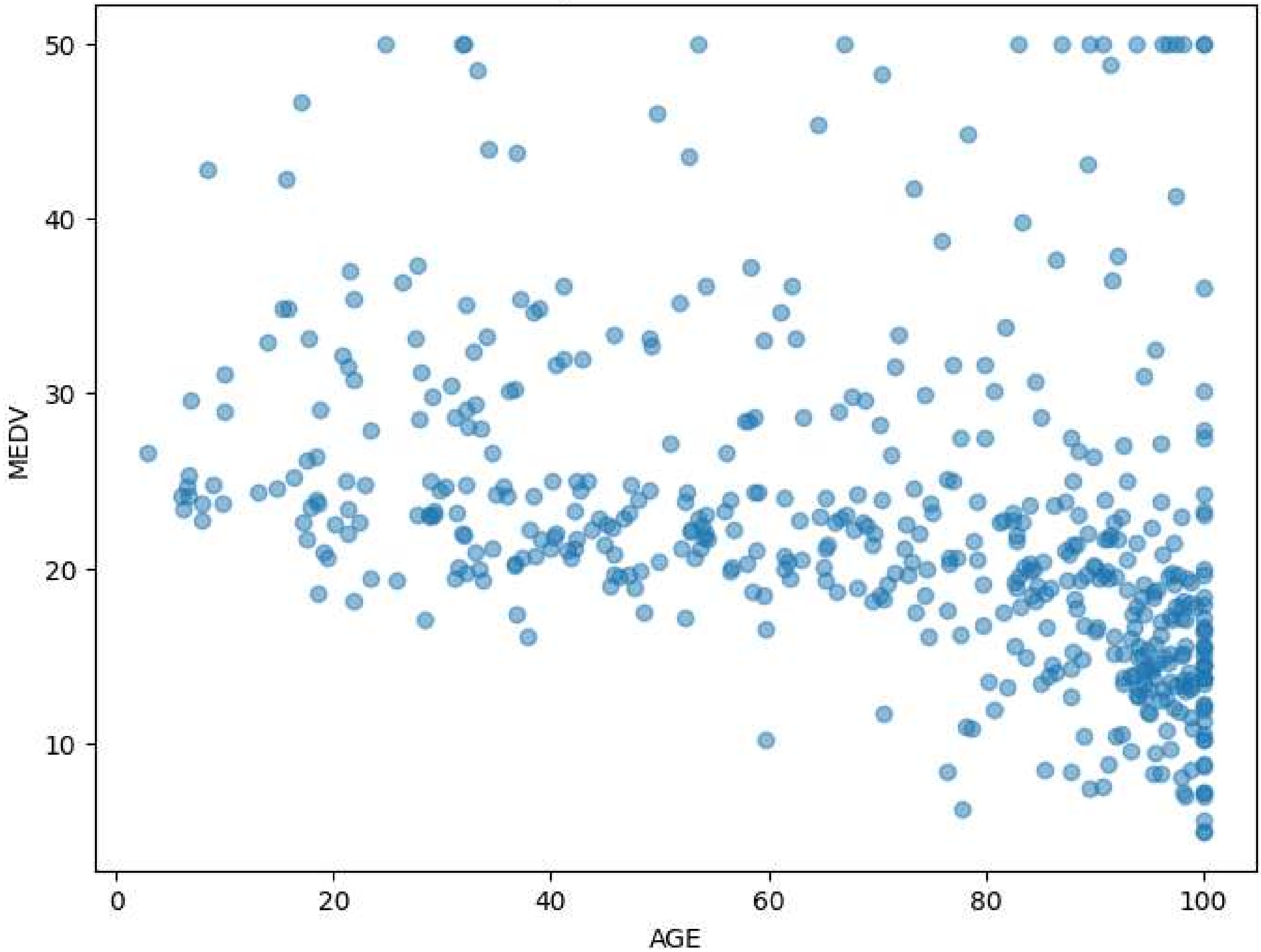
Relationship between NOX and MEDV



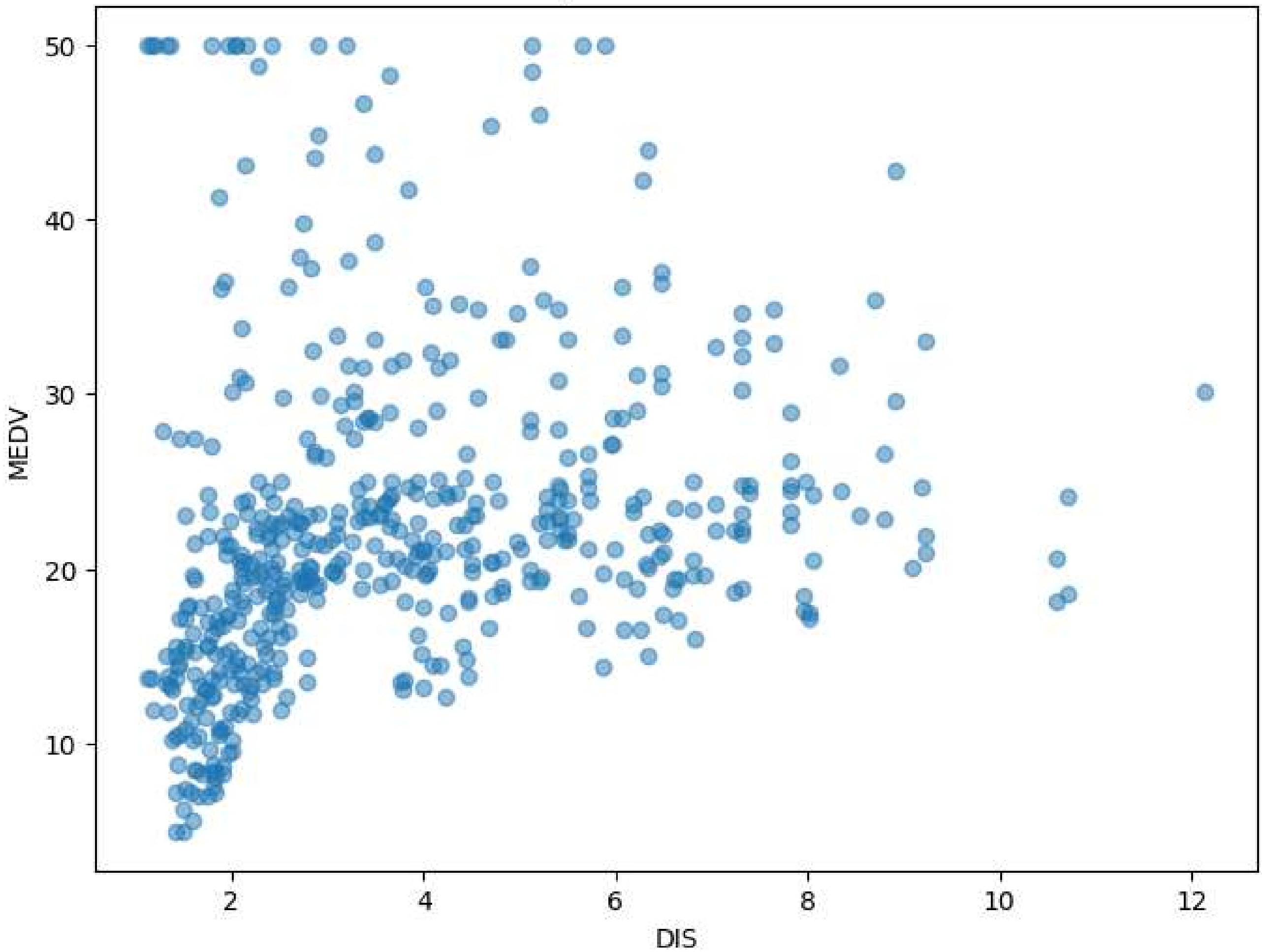
Relationship between RM and MEDV



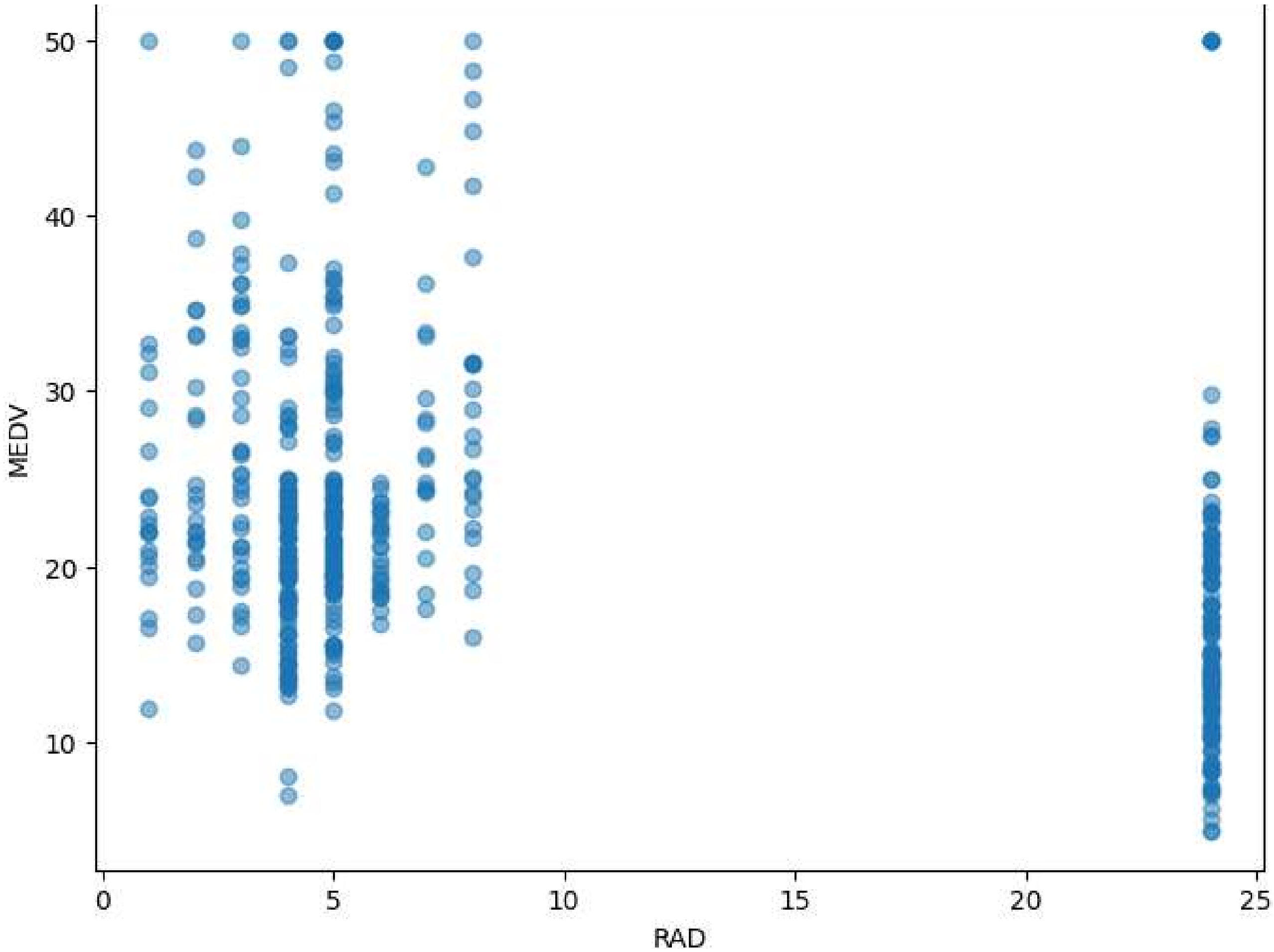
Relationship between AGE and MEDV



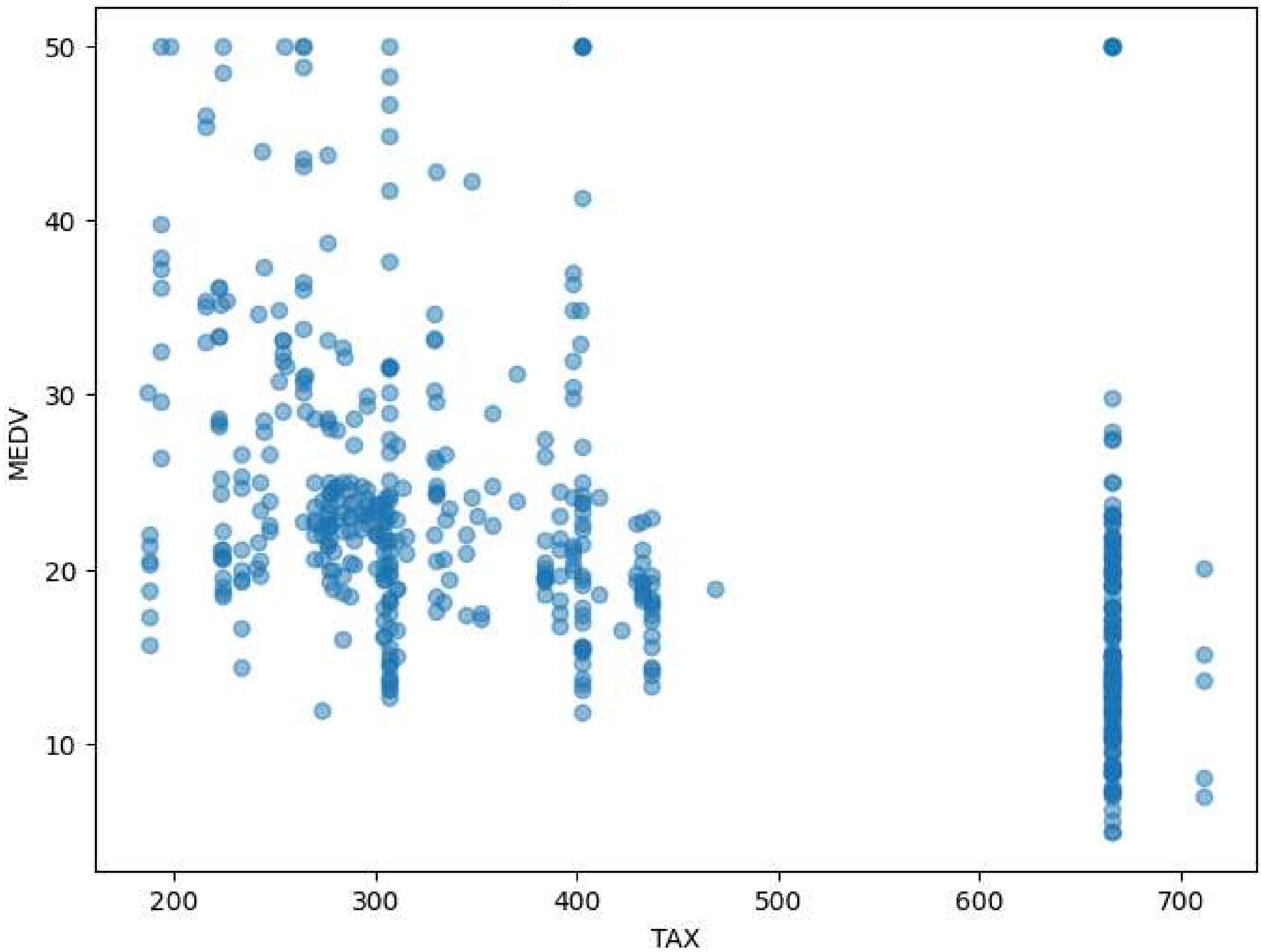
Relationship between DIS and MEDV



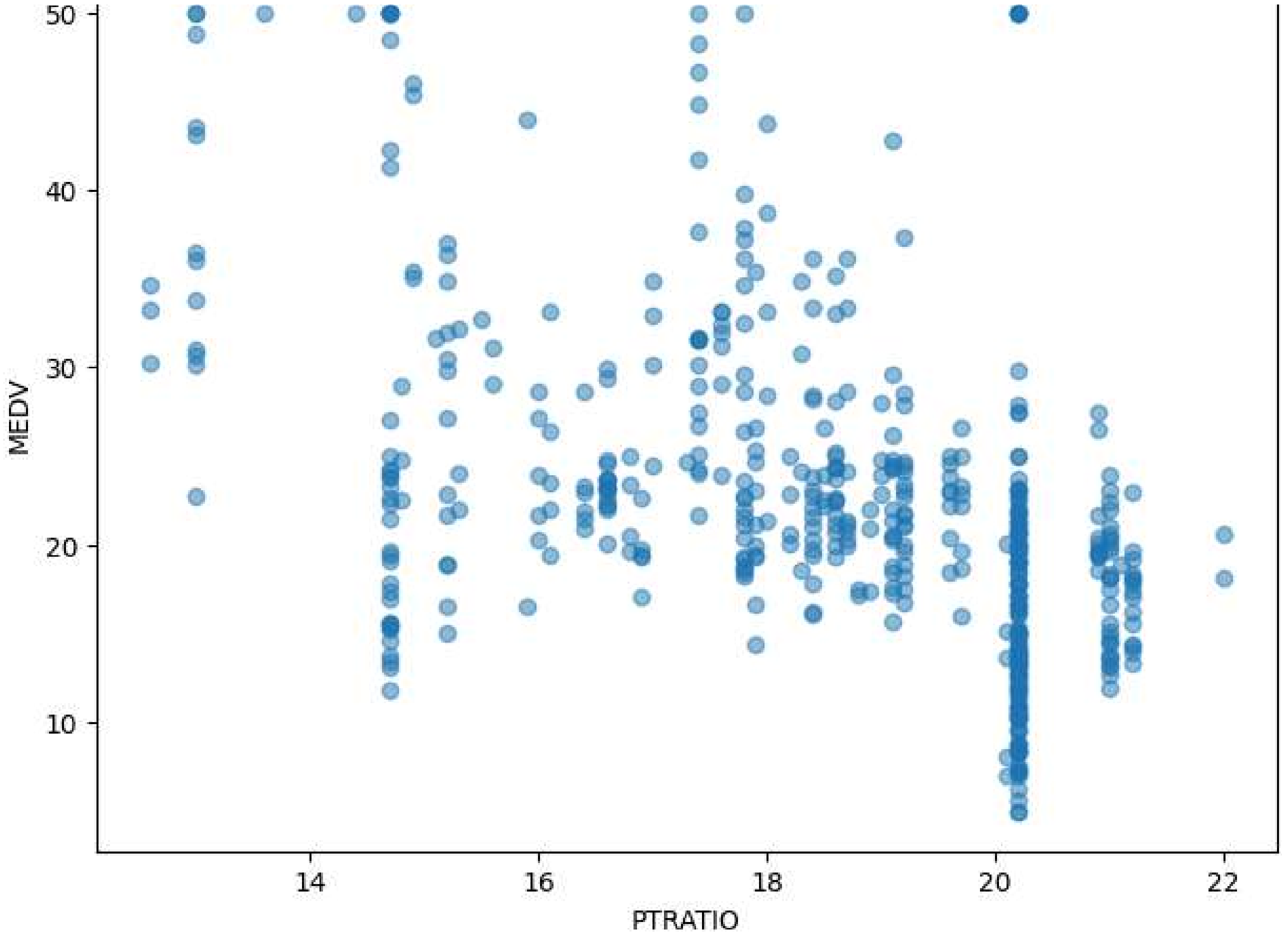
Relationship between RAD and MEDV



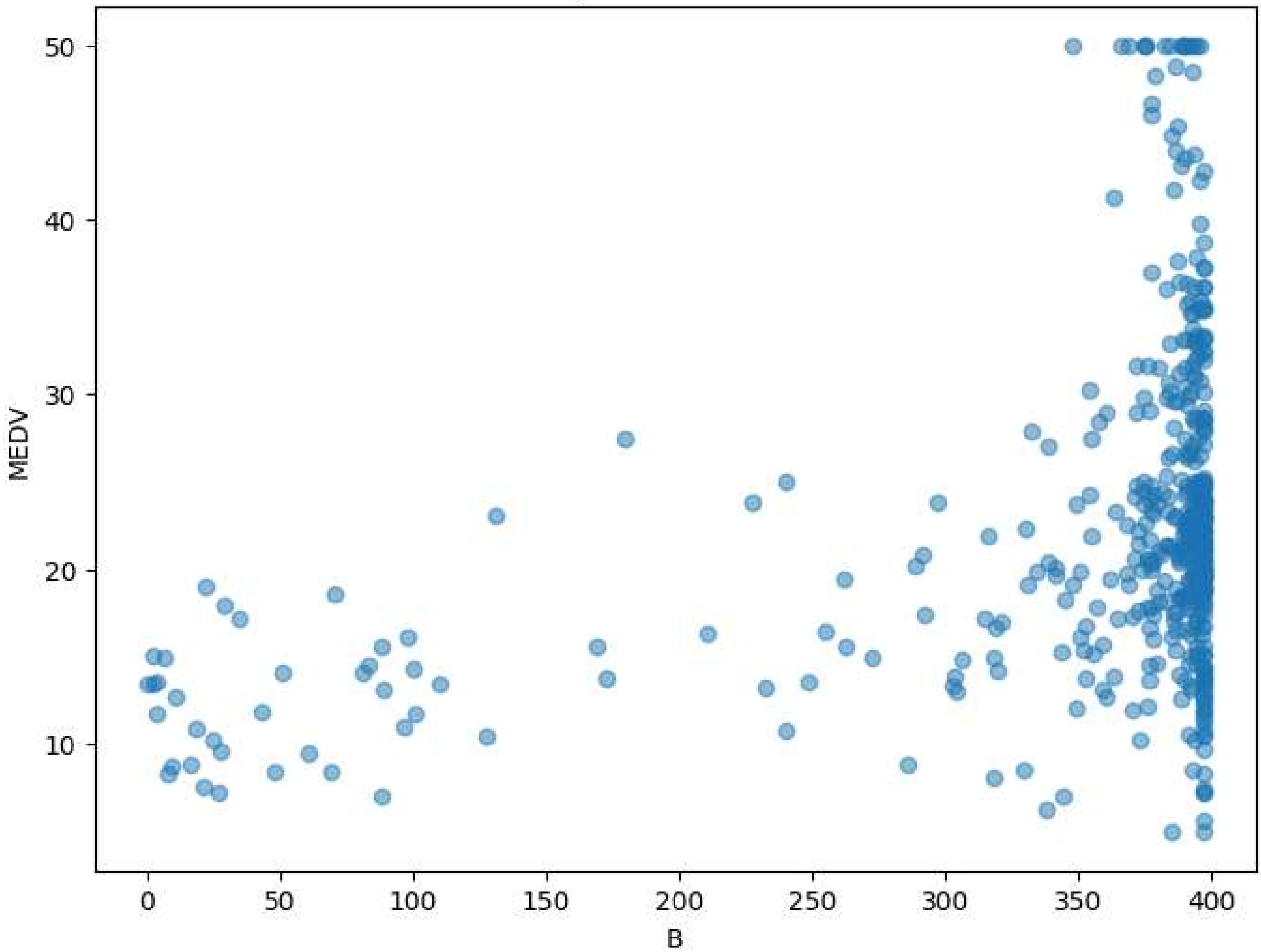
Relationship between TAX and MEDV



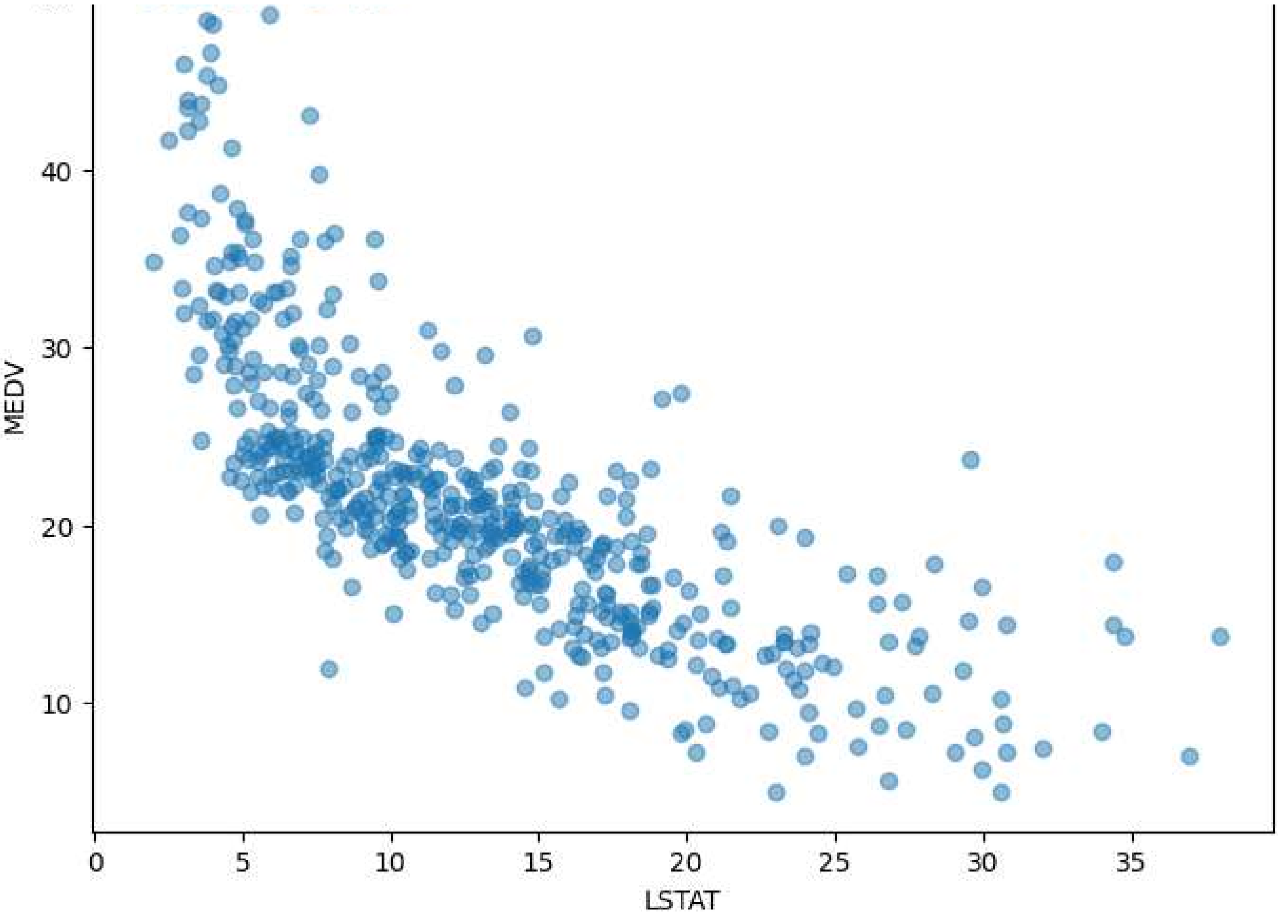
Relationship between PTRATIO and MEDV



Relationship between B and MEDV



Relationship between LSTAT and MEDV



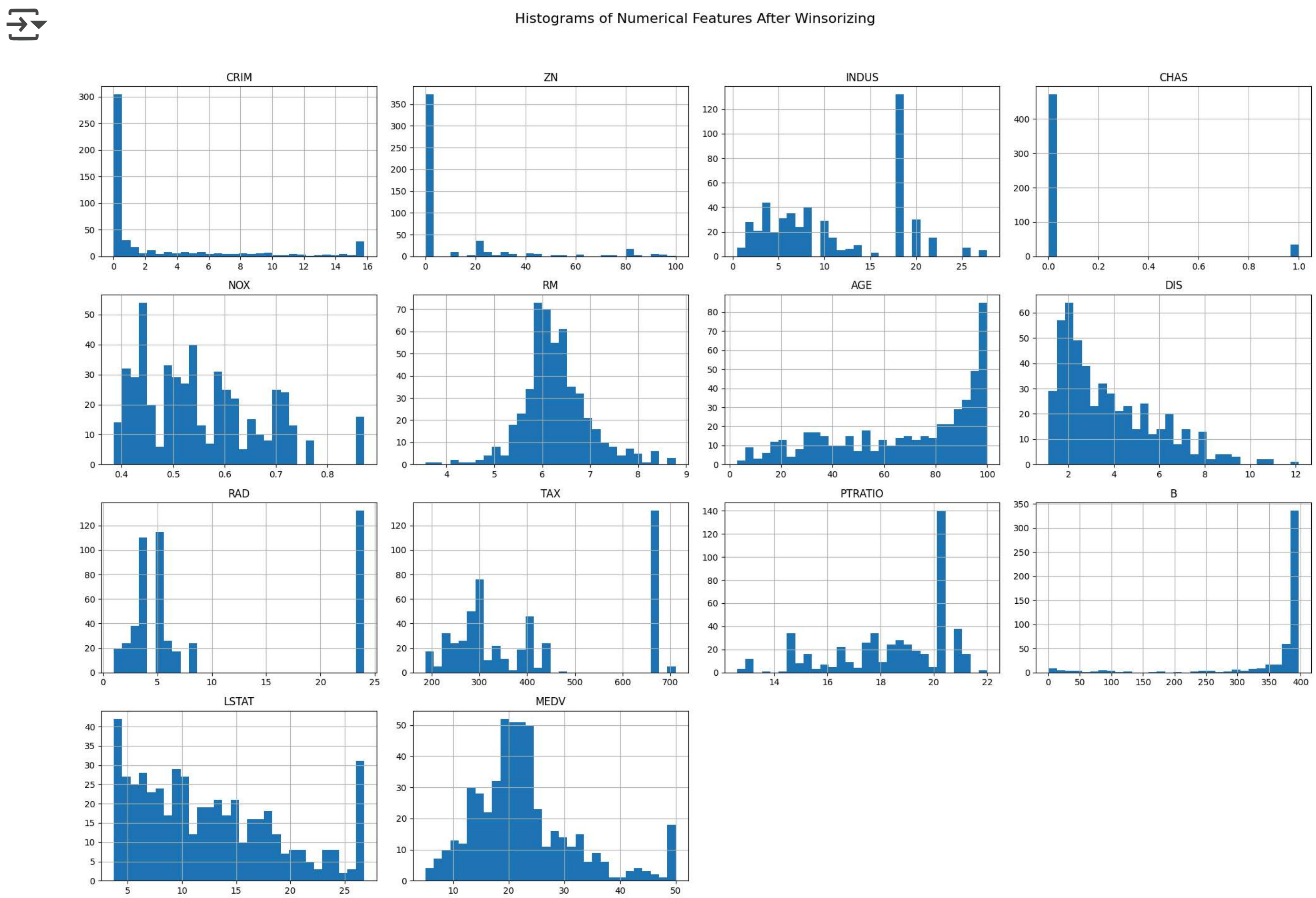
Data cleaning

Subtask:

Handle potential outliers and any remaining issues in the dataset.

Reasoning: Handle outliers in 'CRIM' and 'LSTAT' columns using winsorizing, then re-visualize the distributions to verify the changes.

```
import matplotlib.pyplot as plt
import numpy as np

# Winsorize 'CRIM'
df['CRIM'] = np.clip(df['CRIM'], a_min=df['CRIM'].quantile(0.05), a_max=df['CRIM'].quanti

# Winsorize 'LSTAT'
df['LSTAT'] = np.clip(df['LSTAT'], a_min=df['LSTAT'].quantile(0.05), a_max=df['LSTAT'].qu


# Re-visualize distributions after winsorizing
df.hist(bins=30, figsize=(20, 15))
plt.suptitle('Histograms of Numerical Features After Winsorizing', fontsize=16)
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```



Data preparation

Subtask:

Prepare the data for modeling by scaling numerical features.

Reasoning: Separate features and target, identify numerical features, and scale them using StandardScaler, then replace original numerical features with scaled ones.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# 1. Separate features (X) and target variable (y)
X = df.drop('MEDV', axis=1)
y = df['MEDV']

# 2. Identify numerical features
numerical_features = X.select_dtypes(include=['number']).columns

# 3. Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 4 & 5. Scale numerical features and replace original numerical features
scaler = StandardScaler()
X_train_scaled = X_train.copy()
X_test_scaled = X_test.copy()

X_train_scaled[numerical_features] = scaler.fit_transform(X_train[numerical_features])
X_test_scaled[numerical_features] = scaler.transform(X_test[numerical_features])

display(X_train_scaled.head())
display(X_test_scaled.head())
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS
477	2.650239	-0.500320	1.033237	-0.278089	0.489252	-1.428069	1.028015	-0.802173
15	-0.460349	-0.500320	-0.413160	-0.278089	-0.157233	-0.680087	-0.431199	0.324349
332	-0.588422	1.013271	-0.715218	-0.278089	-1.008723	-0.402063	-1.618599	1.330697
423	0.927494	-0.500320	1.033237	-0.278089	0.489252	-0.300450	0.591681	-0.839240
19	-0.439085	-0.500320	-0.413160	-0.278089	-0.157233	-0.831094	0.033747	-0.005494

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS
173	-0.576080	-0.500320	-1.007111	-0.278089	-0.395412	0.141282	0.555916	-0.545853
274	-0.583716	1.229499	-0.664391	3.595975	-0.931315	0.623942	-1.275254	0.126565
491	-0.573063	-0.500320	2.433163	-0.278089	0.446720	-0.469805	1.081663	-0.911447
72	-0.576110	-0.500320	-0.025421	-0.278089	-1.220532	-0.354079	-2.172957	0.694876
452	0.503937	-0.500320	1.033237	-0.278089	1.331384	-0.026661	0.831307	-0.676503

Model training

Subtask:

Train a linear regression model.

Reasoning: Train a linear regression model on the scaled training data and make predictions on the scaled test data.

```
from sklearn.linear_model import LinearRegression

# Initialize the model
model = LinearRegression()

# Train the model
model.fit(X_train_scaled, y_train)

# Make predictions
y_pred = model.predict(X_test_scaled)
```

Model evaluation

Subtask:

Evaluate the linear regression model's performance.

Reasoning: Evaluate the linear regression model's performance using R-squared, RMSE, and MAE.

```
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
import numpy as np

# Calculate evaluation metrics
r2 = r2_score(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
mae = mean_absolute_error(y_test, y_pred)

# Print the metrics
print(f"R-squared: {r2:.2f}")
print(f"RMSE: {rmse:.2f}")
print(f"MAE: {mae:.2f}")
```

```
R-squared: 0.70
RMSE: 4.68
MAE: 3.14
```

Model optimization

Subtask:

Model Optimization: Explore alternative regression models and hyperparameter tuning.

Reasoning: Train Ridge, Lasso, and RandomForestRegressor models with default hyperparameters, evaluate their performance, and optionally tune hyperparameters for the best-performing model.

```
from sklearn.linear_model import Ridge, Lasso
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
import numpy as np

# Initialize models
ridge_model = Ridge()
lasso_model = Lasso()
rf_model = RandomForestRegressor()

# Train models
ridge_model.fit(X_train_scaled, y_train)
lasso_model.fit(X_train_scaled, y_train)
rf_model.fit(X_train_scaled, y_train)

# Make predictions
ridge_pred = ridge_model.predict(X_test_scaled)
lasso_pred = lasso_model.predict(X_test_scaled)
rf_pred = rf_model.predict(X_test_scaled)

def evaluate_model(y_true, y_pred):
    r2 = r2_score(y_true, y_pred)
    rmse = np.sqrt(mean_squared_error(y_true, y_pred))
    mae = mean_absolute_error(y_true, y_pred)
    return r2, rmse, mae

# Evaluate models
ridge_r2, ridge_rmse, ridge_mae = evaluate_model(y_test, ridge_pred)
lasso_r2, lasso_rmse, lasso_mae = evaluate_model(y_test, lasso_pred)
rf_r2, rf_rmse, rf_mae = evaluate_model(y_test, rf_pred)

print("Ridge Regression:")
print(f"R-squared: {ridge_r2:.2f}")
print(f"RMSE: {ridge_rmse:.2f}")
print(f"MAE: {ridge_mae:.2f}")

print("\nLasso Regression:")
print(f"R-squared: {lasso_r2:.2f}")
print(f"RMSE: {lasso_rmse:.2f}")
print(f"MAE: {lasso_mae:.2f}")

print("\nRandom Forest Regressor:")
print(f"R-squared: {rf_r2:.2f}")
print(f"RMSE: {rf_rmse:.2f}")
print(f"MAE: {rf_mae:.2f}")

# Compare and select the best model
results = {
    'Linear Regression': (r2, rmse, mae),
    'Ridge': (ridge_r2, ridge_rmse, ridge_mae),
```