In [1]:
```
# Initialize Otter
import otter
grader = otter.Notebook("lab6.ipynb")
```

Lab 6: Fitting Models to Data

In this lab, you will practice using a numerical optimization package `cvxpy` to compute solutions to optimization problems. The example we will use is a linear fit and a quadratic fit.

In [2]:
```
import pandas as pd
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
```

Objectives for Lab 6:

Models and fitting models to data is a common task in data science. In this lab, you will practice fitting models to data. The models you will fit are:

- Linear fit
- Normal distribution

Boston Housing Dataset

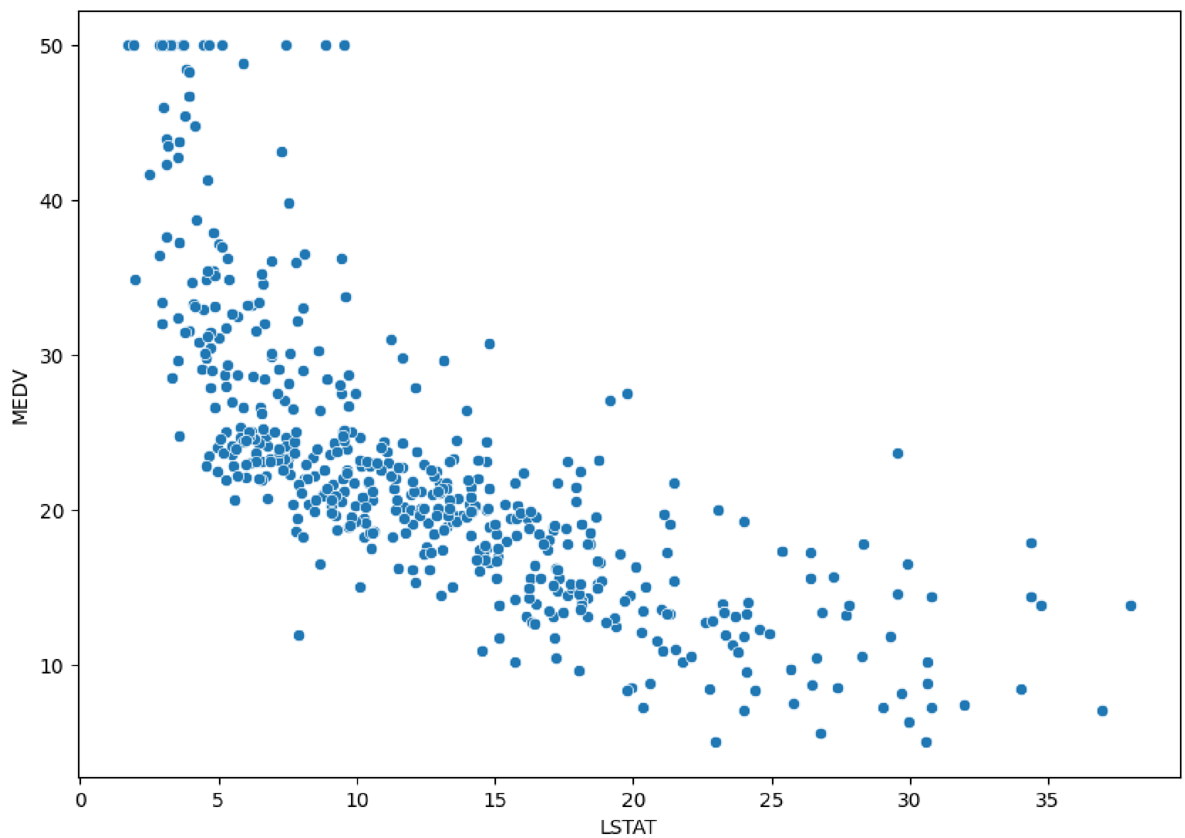
In [3]:
```
data_url = "http://lib.stat.cmu.edu/datasets/boston"
raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)
data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :3]])
from urllib.request import urlopen
html = urlopen(data_url).read()
text = html.decode("utf-8")  # Decode the HTML content into text

lines = text.splitlines()[7:21]

features = []
for line in lines:
    line_strings = line.split()
    if line_strings:
        first_string = line_strings[0]
        features.append(first_string)

housing = pd.DataFrame(data, columns = features)
housing.head()
```

Out[3]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90

In [4]:

```
fig, ax = plt.subplots(figsize=(10, 7))
sns.scatterplot(x='LSTAT', y='MEDV', data=housing)
plt.show()
```



The model for the relationship between the response variable MEDV (y) and predictor variable LSTAT (u) is

$$y_i = \beta_0 + \beta_1 u_i + \epsilon_i,$$

where ϵ_i is random noise.

In order to fit the linear model to data, we minimize the sum of squared errors of all observations, $i = 1, 2, \ldots, n$.

$$\min_{\beta} \sum_{i=1}^{n} (y_i - \beta_0 + \beta_1 u_i)^2 = \min_{\beta} \sum_{i=1}^{n} (y_i - x_i^T \beta)^2 = \min_{\beta} \|y - X\beta\|_2^2$$

where $\beta = (\beta_0, \beta_1)^T$, and $x_i^T = (1, u_i)$. Therefore, $y = (y_1, y_2, \ldots, y_n)^T$ and i-th row of X is x_i^T.

Question 1: Constructing Data Variables

Define y and X from `housing` data.

In [5]:
```
y = housing['MEDV']
X = housing[['LSTAT']]
# X.insert(..., 'intercept', ...)
X.insert(0, 'intercept', 1)
```

In [6]:
```
grader.check("q1")
```

Out[6]:

q1 passed! 🚀

Installing CVXPY

First, install `cvxpy` package by running the following bash command:

In [7]:
```
# !pip install cvxpy
```

Question 2: Fitting Linear Model to Data

Read this example of how cvxpy problem is setup and solved:
https://www.cvxpy.org/examples/basic/least_squares.html

The usage of cvxpy parallels our conceptual understanding of components in an optimization problem:

- `beta2` are the variables β
- `loss2` is sum of squared errors
- `prob2` minimizes the loss by choosing β
- `yhat2` provides estimation of $\hat{y} = x^T \hat{\beta}$

Make sure to extract the data array of data frames (or series) by using `values`: e.g., `X.values`

In [8]:
```
import cvxpy as cp

n, p = X.shape
beta2 = cp.Variable(p)


loss2 = cp.sum_squares(y.values - X.values @ beta2)
```

```
prob2 = cp.Problem(cp.Minimize(loss2))

prob2.solve()

yhat2 = X @ beta2.value
```

In [9]:
```
grader.check("q2")
```

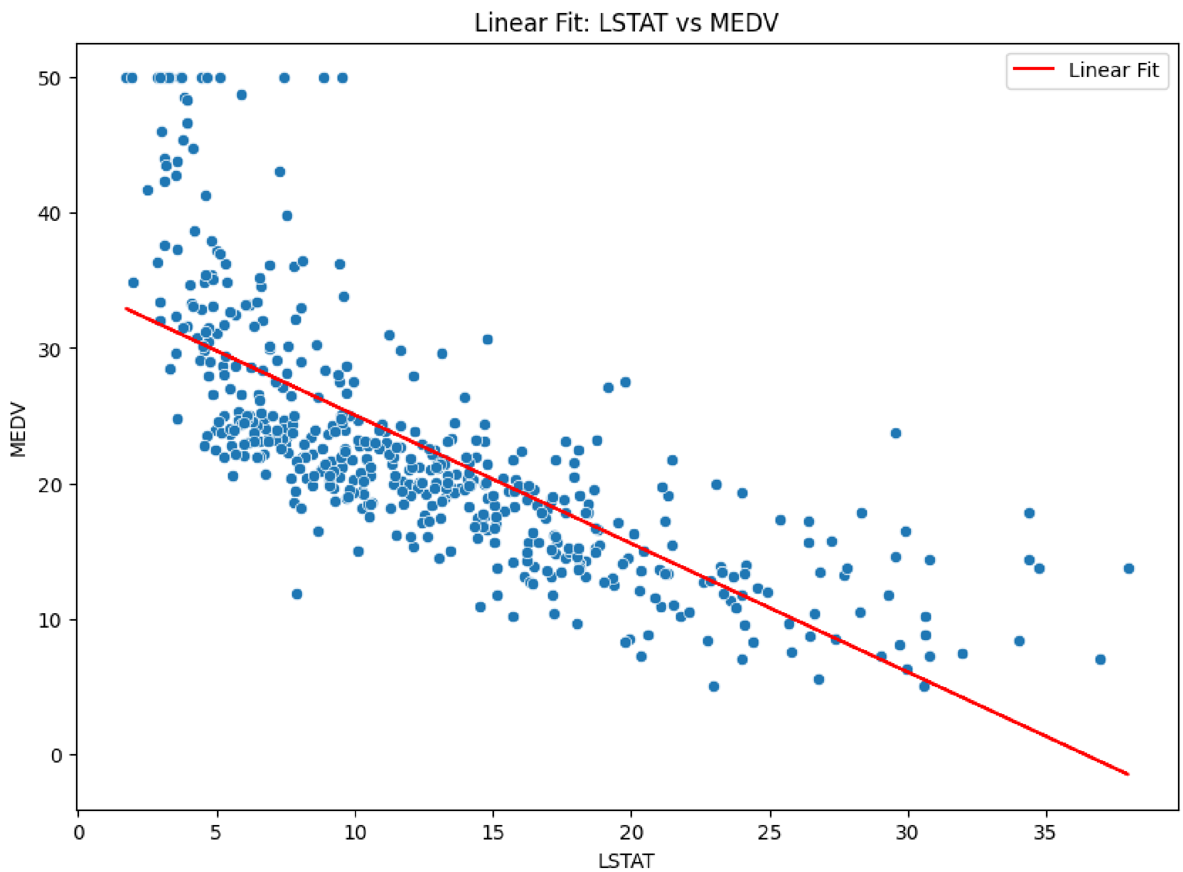
Out[9]:

q2 passed! 🚀

Question 3: Visualizing resulting Linear Fit

Visualize fitted model by plotting `LSTAT` by `MEDV`.

In [10]:
```
fig, ax = plt.subplots(figsize=(10, 7))

sns.scatterplot(x='LSTAT', y='MEDV', data=housing, ax=ax)
ax.plot(housing['LSTAT'], yhat2, color='red', label='Linear Fit')

ax.set_xlabel('LSTAT')
ax.set_ylabel('MEDV')
ax.set_title('Linear Fit: LSTAT vs MEDV')
ax.legend()

plt.show()
```



Question 4: Fitting Quadratic Model to Data

Add a column of squared `LSTAT` values to `X`. The new model is,

Then, fit a quadratic model to data.

In [11]:
```
X2 = X.copy()
X2.insert(2, 'LSTAT^2', X2['LSTAT']**2)


n, p = X2.shape


beta4 = cp.Variable(p)
loss4 = cp.sum_squares(y.values - X2.values @ beta4)
prob4 = cp.Problem(cp.Minimize(loss4))

prob4.solve()

yhat4 = X2 @ beta4.value
```

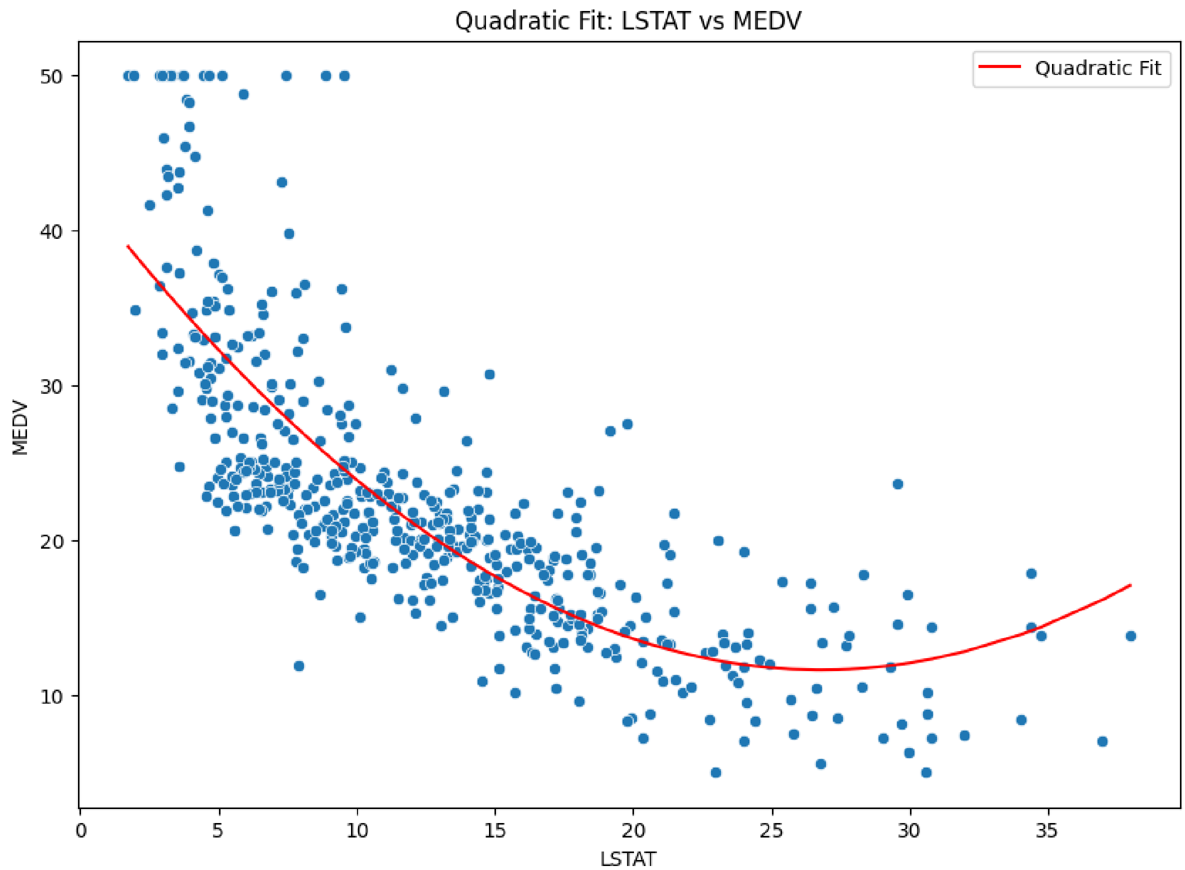
In [12]:
```
grader.check("q4a")
```

Out[12]:

q4a passed! 💯

Visualize quadratic fit:

In [13]:
```
fig, ax = plt.subplots(figsize=(10, 7))

sns.scatterplot(x='LSTAT', y='MEDV', data=housing, ax=ax)


x_vals=X2.iloc[X2['LSTAT'].argsort()]
y_vals = yhat4[X2['LSTAT'].argsort()]
ax.plot(x_vals['LSTAT'], y_vals, color='red', label='Quadratic Fit')

ax.set_xlabel('LSTAT')
ax.set_ylabel('MEDV')
ax.set_title('Quadratic Fit: LSTAT vs MEDV')

ax.legend()

plt.show()
```



To double-check your work, the cell below will rerun all of the autograder tests.

In [14]:
```
grader.check_all()
```

Out[14]:
```
q1 results: All test cases passed!

q2 results: All test cases passed!

q4a results: All test cases passed!
```

Submission

1. Save file to confirm all changes are on disk
2. Run *Kernel > Restart & Run All* to execute all code from top to bottom
3. Save file again to write any new output to disk
4. Select *File > Save and export Notebook as > HTML*.
5. Open in Google Chrome and print to PDF.
6. Submit to Gradescope