Preprocessing

In [29]:

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

Step 1:

Import the boston house dataset from `sklearn.datasets` . Create train and test datasets. Check the data description and familiarize yourself with the data.

In [4]:

```
from sklearn.datasets import load_boston
boston = load_boston()
from sklearn.model_selection import train_test_split
X, y = boston.data, boston.target
X_train, X_test, y_train, y_test = train_test_split(
    X, y, random_state=0)
```

In [5]:

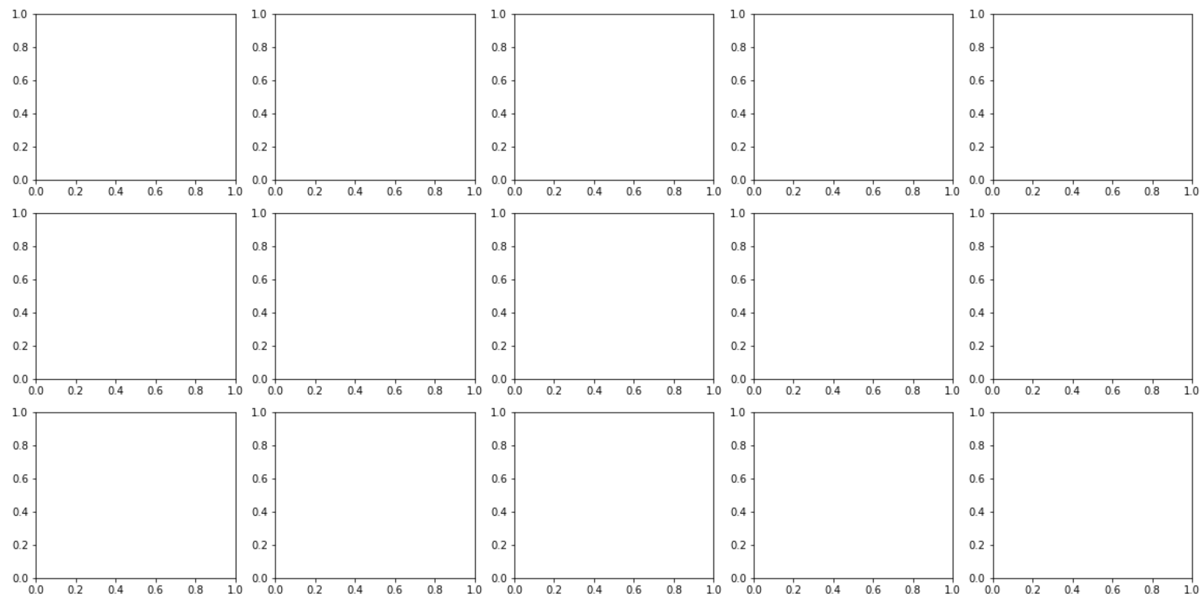
```
print(boston.DESCR)
```

```
.. _boston_dataset:

Boston house prices dataset
---------------------------

**Data Set Characteristics:**

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive. Median Va
lue (attribute 14) is usually the target.

    :Attribute Information (in order):
        - CRIM     per capita crime rate by town
        - ZN       proportion of residential land zoned for lots over 2
5,000 sq.ft.
        - INDUS    proportion of non-retail business acres per town
        - CHAS     Charles River dummy variable (= 1 if tract bounds ri
ver; 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 19
40
        - DIS      weighted distances to five Boston employment centres
        - 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 blac
ks by town
        - LSTAT    % lower status of the population
        - MEDV     Median value of owner-occupied homes in $1000's

    :Missing Attribute Values: None

    :Creator: Harrison, D. and Rubinfeld, D.L.

This is a copy of UCI ML housing dataset.
https://archive.ics.uci.edu/ml/machine-learning-databases/housing/


This dataset was taken from the StatLib library which is maintained at
Carnegie Mellon University.

The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedoni
c
prices and the demand for clean air', J. Environ. Economics & Managemen
t,
vol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Regression diagn
ostics
...', Wiley, 1980.   N.B. Various transformations are used in the table
on
pages 244-261 of the latter.

The Boston house-price data has been used in many machine learning pape
rs that address regression
problems.
```

```
.. topic:: References

   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influe
ntial Data and Sources of Collinearity', Wiley, 1980. 244-261.
   - Quinlan,R. (1993). Combining Instance-Based and Model-Based Learni
ng. In Proceedings on the Tenth International Conference of Machine Lea
rning, 236-243, University of Massachusetts, Amherst. Morgan Kaufmann.
```

In [7]:

```python
fig, axes = plt.subplots(3, 5, figsize=(20, 10))
```



In [12]:

```python
axes.ravel()
```

Out[12]:

```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x1a1f9819e8>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x1a1f98fcc0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x11098cef0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x1a1fc58160>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x1a1fc8b390>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x1a1fcbf5c0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x1a1fcf7320>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x1a1fd296d8>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x1a1fd5aa90>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x1a1fd8dda0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x1a1fdcb160>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x1a1fdfd4e0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x1a1fe30860>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x1a1fe62be0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x1a1fe94f60>],
      dtype=object)
```

In [ ]:

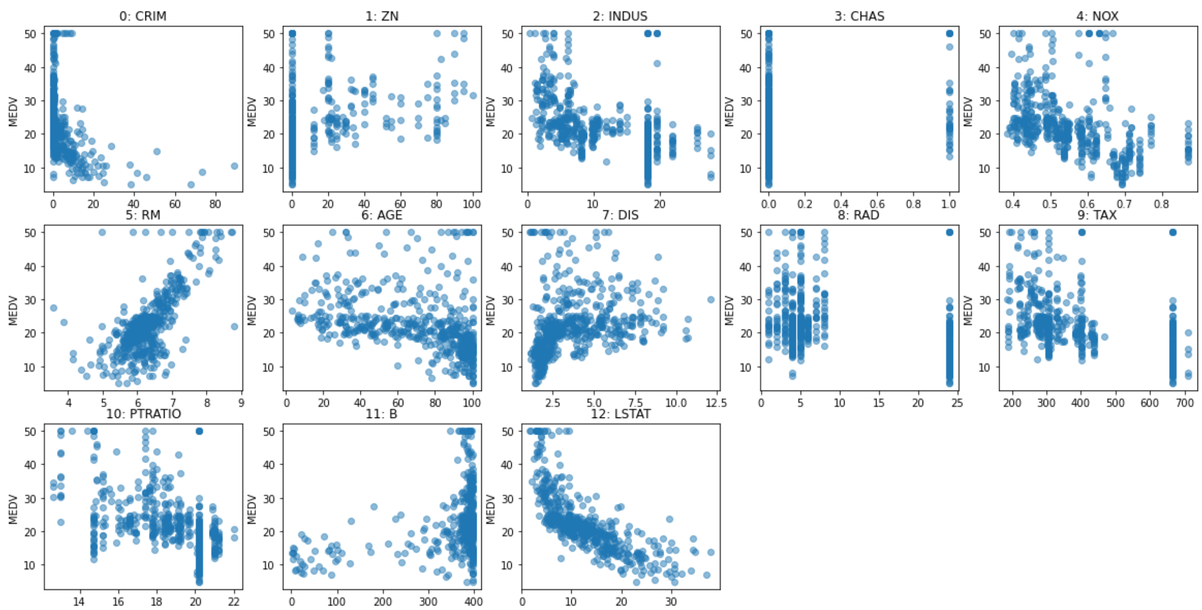
```
dir(axes[0][0])
```

Step 2:

Create a scatter plot of each attribute with the mean house price.

In [6]:

```
fig, axes = plt.subplots(3, 5, figsize=(20, 10))
for i, ax in enumerate(axes.ravel()):
    if i > 12:
        ax.set_visible(False)
        continue
    ax.plot(X[:, i], y, 'o', alpha=.5)
    ax.set_title("{}: {}".format(i, boston.feature_names[i]))
    ax.set_ylabel("MEDV")
```

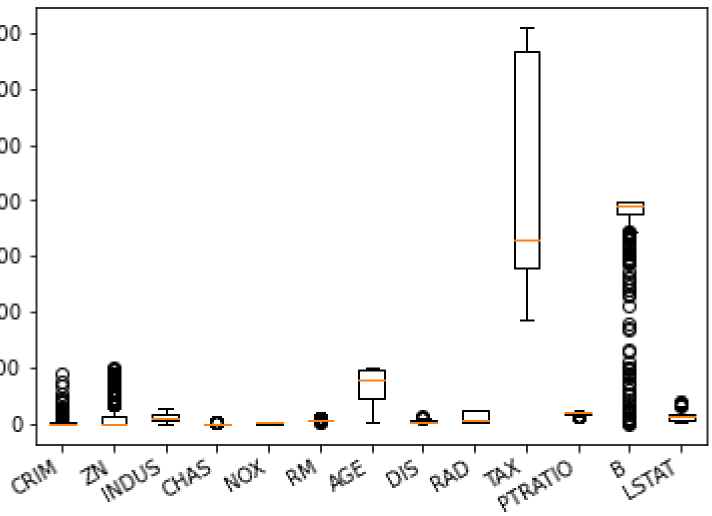


Step 3:

Create a box plot of all attributes.

In [27]:

```
plt.boxplot(X)
plt.xticks(np.arange(1, X.shape[1] + 1),
           boston.feature_names, rotation=30, ha="right"); #horizontalalignment or
 ha
```



Step 4:

Use `StandardScaler()` to scale the trainin data set.

In [28]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
```

Step 5:

Use `KNeighborsRegressor()` to fit both not-scaled and scaled datasets. Check the scores of test dataset for both models.

In [29]:

```
from sklearn.neighbors import KNeighborsRegressor
knr = KNeighborsRegressor().fit(X_train, y_train)
knr.score(X_train, y_train)
```

Out[29]:

```
0.7066272660046313
```

In [30]:

```
knr.score(X_test, y_test)
```

Out[30]:

```
0.4616380924610112
```

In [31]:

```
knr_scaled = KNeighborsRegressor().fit(X_train_scaled, y_train)
knr_scaled.fit(X_train_scaled, y_train)
knr_scaled.score(X_train_scaled, y_train)
```

Out[31]:

```
0.849576948978109
```

In [32]:

```
X_test_scaled = scaler.transform(X_test)
knr_scaled.score(X_test_scaled, y_test)
```

Out[32]:

```
0.606952770711171
```

Step 6:

Repeat Step 5 for `RandomForestRegressor` .

In [33]:

```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators=100, random_state=0)
rf.fit(X_train, y_train)
rf.score(X_test, y_test)
```

Out[33]:

```
0.7952684623500126
```

In [34]:

```
rf_scaled = RandomForestRegressor(n_estimators=100, random_state=0)
rf_scaled.fit(X_train_scaled, y_train)
rf_scaled.score(X_test_scaled, y_test)
```

Out[34]:

```
0.7953897811917161
```