Boston-House-Prediction

March 10, 2019

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import ShuffleSplit
```

In [6]:
```
# Load the Boston housing dataset
data = pd.read_csv('data.csv')
cols=list(data)
prices = data['MDEV']
features = data.drop('MDEV', axis = 1)

# Success
print("Boston housing dataset has {} data points with {} variables each.".format(*data.s
```

```
Boston housing dataset has 489 data points with 4 variables each.
```

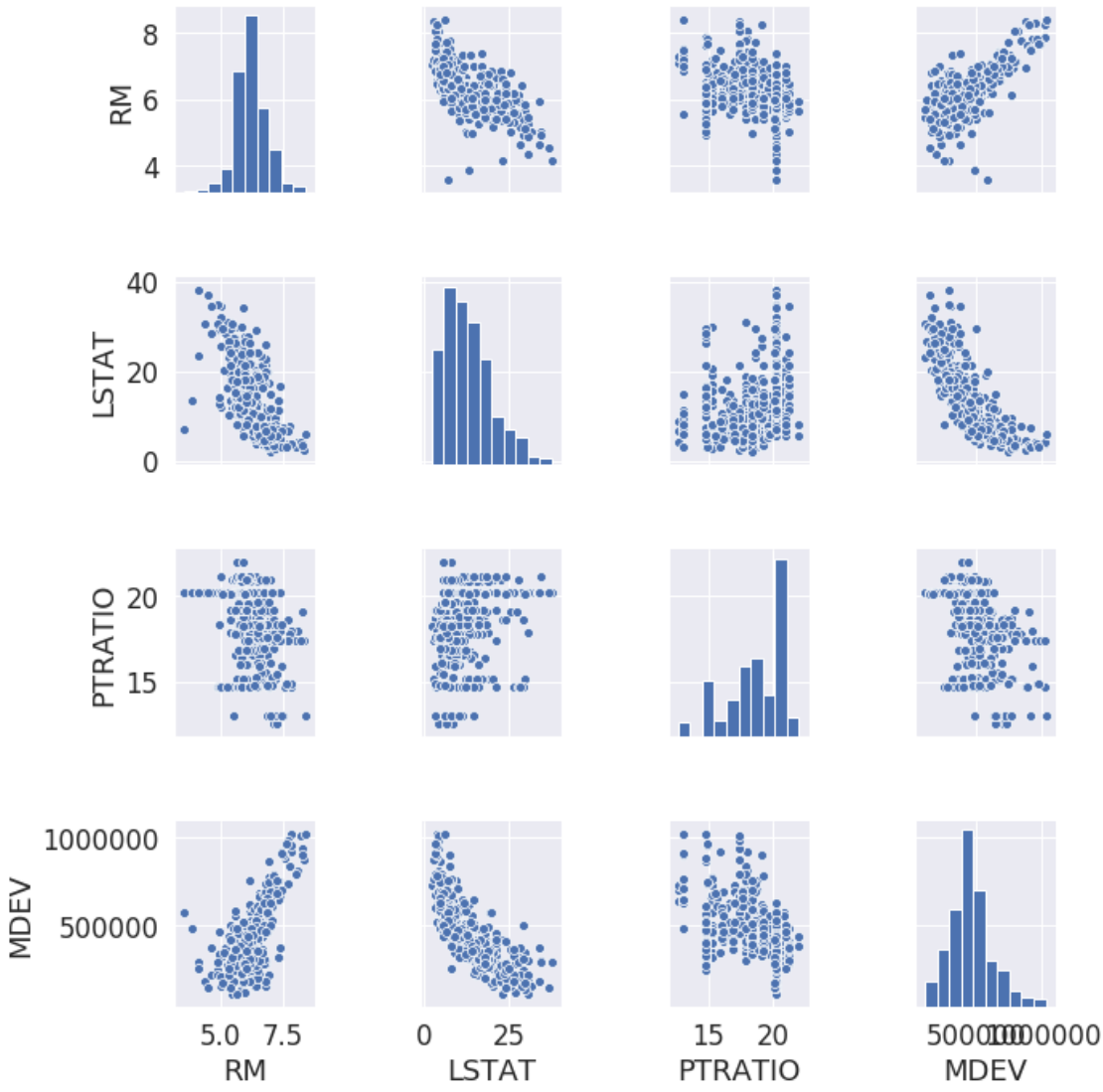
In [7]:
```
# Minimum price of the data
minimum_price = np.amin(prices)

# Maximum price of the data
maximum_price = np.amax(prices)

# Mean price of the data
mean_price = np.mean(prices)

# Median price of the data
median_price = np.median(prices)

# Standard deviation of prices of the data
std_price = np.std(prices)

# Show the calculated statistics
print("Statistics for Boston housing dataset:\n")
print("Minimum price: ${}".format(minimum_price))
print("Maximum price: ${}".format(maximum_price))
print("Mean price: ${}".format(mean_price))
```

```
        print("Median price ${}".format(median_price))
        print("Standard deviation of prices: ${}".format(std_price))
```

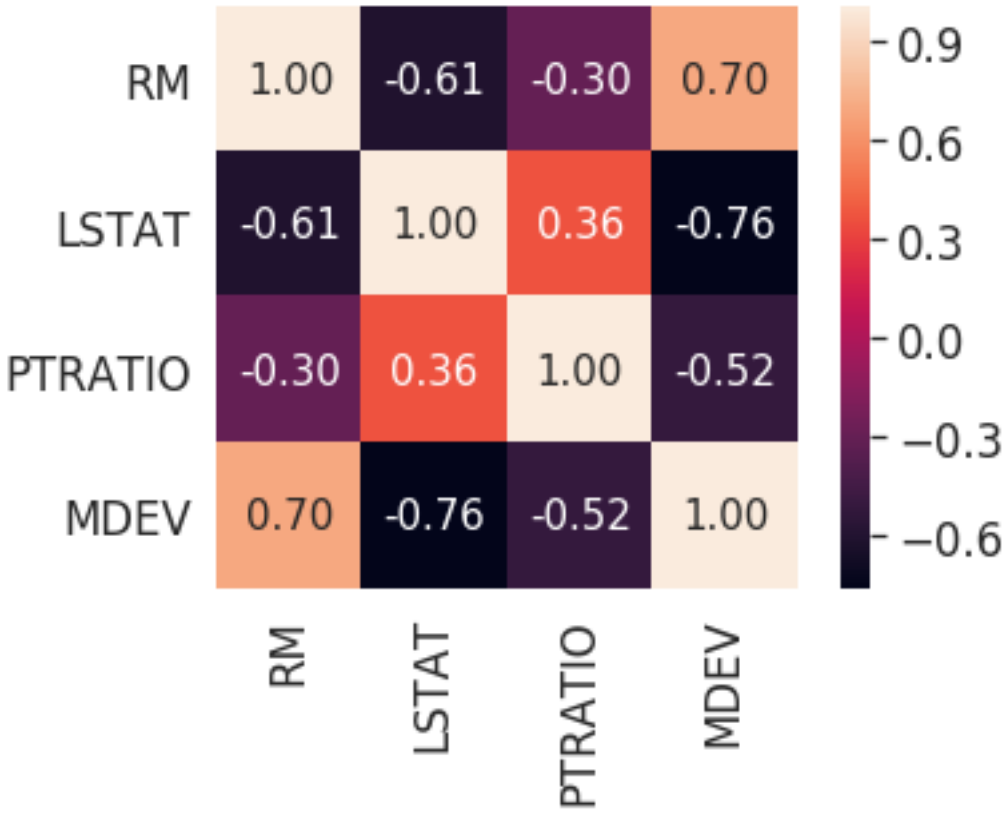
```
Statistics for Boston housing dataset:

Minimum price: $105000.0
Maximum price: $1024800.0
Mean price: $454342.9447852761
Median price $438900.0
Standard deviation of prices: $165171.13154429474
```

In [8]:
```
# Calculate and show pairplot
sns.pairplot(data, size=2.5)
plt.tight_layout()
```

```
/home/pc-3/.local/lib/python3.6/site-packages/seaborn/axisgrid.py:2065: UserWarning: The `size`
  warnings.warn(msg, UserWarning)
```



In [9]:
```
# Calculate and show correlation matrix
cm = np.corrcoef(data.values.T)
sns.set(font_scale=1.5)
hm = sns.heatmap(cm,
                 cbar=True,
                 annot=True,
                 square=True,
                 fmt='.2f',
                 annot_kws={'size': 15},
                 yticklabels=cols,
                 xticklabels=cols)
```



In [11]:
```
# Import 'r2_score'

from sklearn.metrics import r2_score

def performance_metric(y_true, y_predict):
    """ Calculates and returns the performance score between
        true (y_true) and predicted (y_predict) values based on the metric chosen. """

    score = r2_score(y_true, y_predict)

    # Return the score
    return score
```

In [12]:
```
# Import 'train_test_split'
from sklearn.model_selection import train_test_split

# Shuffle and split the data into training and testing subsets
x_train, x_test, y_train, y_test = train_test_split(features, prices, test_size=0.2, ra

# Success
print("Training and testing split was successful.")
```

```
Training and testing split was successful.
```

In [ ]: