

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

```
In [2]: # Importing the Boston Housing dataset
from sklearn.datasets import load_boston
boston = load_boston()
```

/home/ubuntu/anaconda3/lib/python3.9/site-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function load\_boston is deprecated; `load\_boston` is deprecated in 1.0 and will be removed in 1.2.

The Boston housing prices dataset has an ethical problem. You can refer to the documentation of this function for further details.

The scikit-learn maintainers therefore strongly discourage the use of this dataset unless the purpose of the code is to study and educate about ethical issues in data science and machine learning.

In this special case, you can fetch the dataset from the original source::

```
import pandas as pd
import numpy as np
```

```
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, :2]])
target = raw_df.values[1::2, 2]
```

Alternative datasets include the California housing dataset (i.e. :func:`~sklearn.datasets.fetch\_california\_housing`) and the Ames housing dataset. You can load the datasets as follows::

```
from sklearn.datasets import fetch_california_housing
housing = fetch_california_housing()
```

for the California housing dataset and::

```
from sklearn.datasets import fetch_openml
housing = fetch_openml(name="house_prices", as_frame=True)
```

for the Ames housing dataset.

```
warnings.warn(msg, category=FutureWarning)
```

```
In [3]: # Initializing the dataframe
data = pd.DataFrame(boston.data)
```

```
In [5]: data.columns = boston.feature_names
data.head()
```

```
Out[5]:
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LST.
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	4.
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	9.
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	4.
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	2.
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	5.

```
In [6]: data['PRICE'] = boston.target
```

```
In [7]: data.isnull().sum()
```

```
Out[7]: 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
PRICE       0
dtype: int64
```

```
In [8]: x = data.drop(['PRICE'], axis = 1)
y = data['PRICE']
```

```
In [10]: from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size =0.2, random
```

```
In [11]: import sklearn
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
model=lm.fit(xtrain, ytrain)
```

```
In [12]: ytrain_pred = lm.predict(xtrain)
ytest_pred = lm.predict(xtest)
```

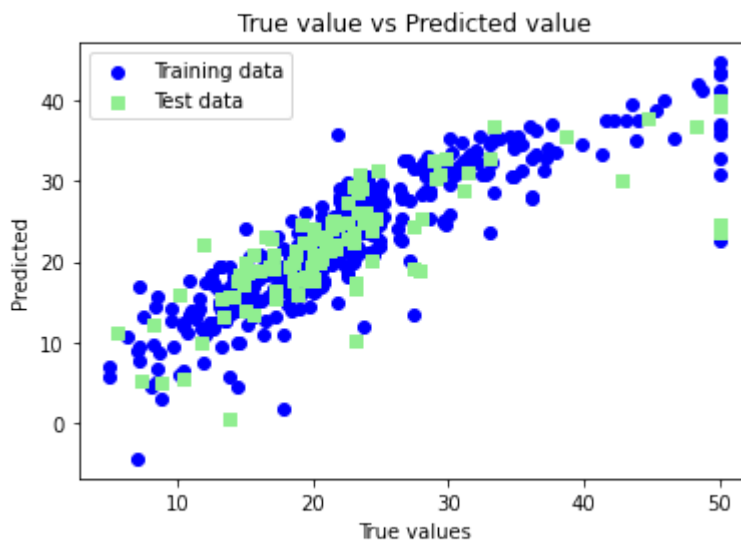
```
In [14]: from sklearn.metrics import mean_squared_error, r2_score
mse = mean_squared_error(ytest, ytest_pred)
print(mse)
mse = mean_squared_error(ytrain_pred, ytrain)
print(mse)
```

```
33.448979997676524
19.326470203585725
```

```
In [15]: mse = mean_squared_error(ytest, ytest_pred)
print(mse)
```

33.448979997676524

```
In [17]: plt.scatter(ytrain ,ytrain_pred,c='blue',marker='o',label='Training data')
plt.scatter(ytest,ytest_pred ,c='lightgreen',marker='s',label='Test data')
plt.xlabel('True values')
plt.ylabel('Predicted')
plt.title("True value vs Predicted value")
plt.legend(loc= 'upper left')
#plt.hlines(y=0,xmin=0,xmax=50)
plt.plot()
plt.show()
```



```
In [ ]:
```