Exploratory Data Analysis and Best fit model

February 17, 2022

[15]:
```
import pandas as pd
from sklearn.datasets import load_boston
pd.options.display.float_format = '{:,.2f}'.format
import seaborn as sns
import matplotlib.pyplot as plt
dataset = load_boston()
print("[INFO] keys : {}".format(dataset.keys()))
```

```
[INFO] keys : dict_keys(['data', 'target', 'feature_names', 'DESCR',
'filename'])
```

[2]:
```
print("[INFO] features shape : {}".format(dataset.data.shape))
print("[INFO] target shape   : {}".format(dataset.target.shape))
print("[INFO] feature names")
print(dataset.feature_names)
```

```
[INFO] features shape : (506, 13)
[INFO] target shape   : (506,)
[INFO] feature names
['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO'
 'B' 'LSTAT']
```

[3]:
```
print("[INFO] dataset summary")
print(dataset.DESCR)
```

```
[INFO] dataset summary
.. _boston_dataset:

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

**Data Set Characteristics:**

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive. Median Value
(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 25,000
sq.ft.
        - INDUS    proportion of non-retail business acres per town
        - CHAS     Charles River dummy variable (= 1 if tract bounds river; 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 1940
        - 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 black people
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. 'Hedonic
prices and the demand for clean air', J. Environ. Economics & Management,
vol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Regression diagnostics
…', 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 papers that
address regression
problems.

.. topic:: References

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

[4]:
```python
df = pd.DataFrame(dataset.data)
print("[INFO] df type : {}".format(type(df)))
print("[INFO] df shape: {}".format(df.shape))
print(df.head())
```

```
[INFO] df type : <class 'pandas.core.frame.DataFrame'>
[INFO] df shape: (506, 13)
     0     1    2    3    4    5     6    7    8      9    10     11   12
0 0.01 18.00 2.31 0.00 0.54 6.58 65.20 4.09 1.00 296.00 15.30 396.90 4.98
1 0.03  0.00 7.07 0.00 0.47 6.42 78.90 4.97 2.00 242.00 17.80 396.90 9.14
2 0.03  0.00 7.07 0.00 0.47 7.18 61.10 4.97 2.00 242.00 17.80 392.83 4.03
3 0.03  0.00 2.18 0.00 0.46 7.00 45.80 6.06 3.00 222.00 18.70 394.63 2.94
4 0.07  0.00 2.18 0.00 0.46 7.15 54.20 6.06 3.00 222.00 18.70 396.90 5.33
```

[5]:
```python
df.columns = dataset.feature_names
print(df.head())
```

```
   CRIM    ZN  INDUS  CHAS  NOX   RM   AGE  DIS  RAD    TAX  PTRATIO      B  \
0  0.01 18.00   2.31  0.00 0.54 6.58 65.20 4.09 1.00 296.00    15.30 396.90
1  0.03  0.00   7.07  0.00 0.47 6.42 78.90 4.97 2.00 242.00    17.80 396.90
2  0.03  0.00   7.07  0.00 0.47 7.18 61.10 4.97 2.00 242.00    17.80 392.83
3  0.03  0.00   2.18  0.00 0.46 7.00 45.80 6.06 3.00 222.00    18.70 394.63
4  0.07  0.00   2.18  0.00 0.46 7.15 54.20 6.06 3.00 222.00    18.70 396.90

   LSTAT
0   4.98
1   9.14
2   4.03
3   2.94
4   5.33
```

[6]:
```python
df["PRICE"] = dataset.target
print(df.head())
```

```
   CRIM    ZN  INDUS  CHAS  NOX   RM   AGE  DIS  RAD    TAX  PTRATIO      B  \
0  0.01 18.00   2.31  0.00 0.54 6.58 65.20 4.09 1.00 296.00    15.30 396.90
1  0.03  0.00   7.07  0.00 0.47 6.42 78.90 4.97 2.00 242.00    17.80 396.90
2  0.03  0.00   7.07  0.00 0.47 7.18 61.10 4.97 2.00 242.00    17.80 392.83
3  0.03  0.00   2.18  0.00 0.46 7.00 45.80 6.06 3.00 222.00    18.70 394.63
4  0.07  0.00   2.18  0.00 0.46 7.15 54.20 6.06 3.00 222.00    18.70 396.90

   LSTAT  PRICE
0   4.98  24.00
1   9.14  21.60
2   4.03  34.70
3   2.94  33.40
4   5.33  36.20
```

1 Correleation

Finding correlation between attributes is a highly useful way to check for patterns in the dataset. Pandas offers three different ways to find correlation between attributes (columns). The output of each of these correlation functions fall within the range [-1, 1].

1 - Positively correlated -1 - Negatively correlated. 0 - Not correlated. To learn more about correlation, please read this wikipedia article. We will use df.corr() function to compute the correlation between attributes and sns.heatmap() function to visualize the correlation matrix.

[16]:
```
print("PEARSON CORRELATION")
print(df.corr(method="pearson"))
sns.heatmap(df.corr(method="pearson"))
plt.savefig("heatmap_pearson.png")
plt.clf()
plt.close()

print("SPEARMAN CORRELATION")
print(df.corr(method="spearman"))
sns.heatmap(df.corr(method="spearman"))
plt.savefig("heatmap_spearman.png")
plt.clf()
plt.close()

print("KENDALL CORRELATION")
print(df.corr(method="kendall"))
sns.heatmap(df.corr(method="kendall"))
plt.savefig("heatmap_kendall.png")
plt.clf()
plt.close()
```

```
PEARSON CORRELATION
         CRIM    ZN  INDUS  CHAS   NOX    RM   AGE   DIS   RAD   TAX  PTRATIO  \
CRIM     1.00 -0.20   0.41 -0.06  0.42 -0.22  0.35 -0.38  0.63  0.58     0.29
ZN      -0.20  1.00  -0.53 -0.04 -0.52  0.31 -0.57  0.66 -0.31 -0.31    -0.39
INDUS    0.41 -0.53   1.00  0.06  0.76 -0.39  0.64 -0.71  0.60  0.72     0.38
CHAS    -0.06 -0.04   0.06  1.00  0.09  0.09  0.09 -0.10 -0.01 -0.04    -0.12
NOX      0.42 -0.52   0.76  0.09  1.00 -0.30  0.73 -0.77  0.61  0.67     0.19
RM      -0.22  0.31  -0.39  0.09 -0.30  1.00 -0.24  0.21 -0.21 -0.29    -0.36
AGE      0.35 -0.57   0.64  0.09  0.73 -0.24  1.00 -0.75  0.46  0.51     0.26
DIS     -0.38  0.66  -0.71 -0.10 -0.77  0.21 -0.75  1.00 -0.49 -0.53    -0.23
RAD      0.63 -0.31   0.60 -0.01  0.61 -0.21  0.46 -0.49  1.00  0.91     0.46
TAX      0.58 -0.31   0.72 -0.04  0.67 -0.29  0.51 -0.53  0.91  1.00     0.46
PTRATIO  0.29 -0.39   0.38 -0.12  0.19 -0.36  0.26 -0.23  0.46  0.46     1.00
B       -0.39  0.18  -0.36  0.05 -0.38  0.13 -0.27  0.29 -0.44 -0.44    -0.18
LSTAT    0.46 -0.41   0.60 -0.05  0.59 -0.61  0.60 -0.50  0.49  0.54     0.37
PRICE   -0.39  0.36  -0.48  0.18 -0.43  0.70 -0.38  0.25 -0.38 -0.47    -0.51

            B  LSTAT  PRICE
```

```
CRIM    -0.39   0.46  -0.39
ZN       0.18  -0.41   0.36
INDUS   -0.36   0.60  -0.48
CHAS     0.05  -0.05   0.18
NOX     -0.38   0.59  -0.43
RM       0.13  -0.61   0.70
AGE     -0.27   0.60  -0.38
DIS      0.29  -0.50   0.25
RAD     -0.44   0.49  -0.38
TAX     -0.44   0.54  -0.47
PTRATIO -0.18   0.37  -0.51
B        1.00  -0.37   0.33
LSTAT   -0.37   1.00  -0.74
PRICE    0.33  -0.74   1.00
SPEARMAN CORRELATION
         CRIM    ZN  INDUS  CHAS   NOX    RM   AGE   DIS   RAD   TAX  PTRATIO  \
CRIM     1.00 -0.57   0.74  0.04  0.82 -0.31  0.70 -0.74  0.73  0.73     0.47
ZN      -0.57  1.00  -0.64 -0.04 -0.63  0.36 -0.54  0.61 -0.28 -0.37    -0.45
INDUS    0.74 -0.64   1.00  0.09  0.79 -0.42  0.68 -0.76  0.46  0.66     0.43
CHAS     0.04 -0.04   0.09  1.00  0.07  0.06  0.07 -0.08  0.02 -0.04    -0.14
NOX      0.82 -0.63   0.79  0.07  1.00 -0.31  0.80 -0.88  0.59  0.65     0.39
RM      -0.31  0.36  -0.42  0.06 -0.31  1.00 -0.28  0.26 -0.11 -0.27    -0.31
AGE      0.70 -0.54   0.68  0.07  0.80 -0.28  1.00 -0.80  0.42  0.53     0.36
DIS     -0.74  0.61  -0.76 -0.08 -0.88  0.26 -0.80  1.00 -0.50 -0.57    -0.32
RAD      0.73 -0.28   0.46  0.02  0.59 -0.11  0.42 -0.50  1.00  0.70     0.32
TAX      0.73 -0.37   0.66 -0.04  0.65 -0.27  0.53 -0.57  0.70  1.00     0.45
PTRATIO  0.47 -0.45   0.43 -0.14  0.39 -0.31  0.36 -0.32  0.32  0.45     1.00
B       -0.36  0.16  -0.29 -0.04 -0.30  0.05 -0.23  0.25 -0.28 -0.33    -0.07
LSTAT    0.63 -0.49   0.64 -0.05  0.64 -0.64  0.66 -0.56  0.39  0.53     0.47
PRICE   -0.56  0.44  -0.58  0.14 -0.56  0.63 -0.55  0.45 -0.35 -0.56    -0.56

            B  LSTAT  PRICE
CRIM    -0.36   0.63  -0.56
ZN       0.16  -0.49   0.44
INDUS   -0.29   0.64  -0.58
CHAS    -0.04  -0.05   0.14
NOX     -0.30   0.64  -0.56
RM       0.05  -0.64   0.63
AGE     -0.23   0.66  -0.55
DIS      0.25  -0.56   0.45
RAD     -0.28   0.39  -0.35
TAX     -0.33   0.53  -0.56
PTRATIO -0.07   0.47  -0.56
B        1.00  -0.21   0.19
LSTAT   -0.21   1.00  -0.85
PRICE    0.19  -0.85   1.00
KENDALL CORRELATION
         CRIM    ZN  INDUS  CHAS   NOX    RM   AGE   DIS   RAD   TAX  PTRATIO  \
```

```
CRIM      1.00 -0.46   0.52  0.03  0.60 -0.21  0.50 -0.54  0.56  0.54     0.31
ZN       -0.46  1.00  -0.54 -0.04 -0.51  0.28 -0.43  0.48 -0.23 -0.29    -0.36
INDUS     0.52 -0.54   1.00  0.08  0.61 -0.29  0.49 -0.57  0.35  0.48     0.34
CHAS      0.03 -0.04   0.08  1.00  0.06  0.05  0.06 -0.07  0.02 -0.04    -0.12
NOX       0.60 -0.51   0.61  0.06  1.00 -0.22  0.59 -0.68  0.43  0.45     0.28
RM       -0.21  0.28  -0.29  0.05 -0.22  1.00 -0.19  0.18 -0.08 -0.19    -0.22
AGE       0.50 -0.43   0.49  0.06  0.59 -0.19  1.00 -0.61  0.31  0.36     0.25
DIS      -0.54  0.48  -0.57 -0.07 -0.68  0.18 -0.61  1.00 -0.36 -0.38    -0.22
RAD       0.56 -0.23   0.35  0.02  0.43 -0.08  0.31 -0.36  1.00  0.56     0.25
TAX       0.54 -0.29   0.48 -0.04  0.45 -0.19  0.36 -0.38  0.56  1.00     0.29
PTRATIO   0.31 -0.36   0.34 -0.12  0.28 -0.22  0.25 -0.22  0.25  0.29     1.00
B        -0.26  0.13  -0.19 -0.03 -0.20  0.03 -0.15  0.17 -0.21 -0.24    -0.04
LSTAT     0.45 -0.39   0.47 -0.04  0.45 -0.47  0.49 -0.41  0.29  0.38     0.33
PRICE    -0.40  0.34  -0.42  0.12 -0.39  0.48 -0.39  0.31 -0.25 -0.41    -0.40

             B  LSTAT  PRICE
CRIM     -0.26   0.45  -0.40
ZN        0.13  -0.39   0.34
INDUS    -0.19   0.47  -0.42
CHAS     -0.03  -0.04   0.12
NOX      -0.20   0.45  -0.39
RM        0.03  -0.47   0.48
AGE      -0.15   0.49  -0.39
DIS       0.17  -0.41   0.31
RAD      -0.21   0.29  -0.25
TAX      -0.24   0.38  -0.41
PTRATIO -0.04   0.33  -0.40
B         1.00  -0.15   0.13
LSTAT    -0.15   1.00  -0.67
PRICE     0.13  -0.67   1.00
```

[17]:

```python
from IPython.display import HTML, display
display(HTML("<table><tr><td><img src='heatmap_kendall.png'></td><td><img␣
↪src='heatmap_pearson.png'></td><td><img src='heatmap_spearman.png'></td></
↪tr></table>"))
```

```
<IPython.core.display.HTML object>
```

[18]:

```python
file_report = "boston_housing.txt"
with open(file_report, "w") as f:
    f.write("Features shape : {}".format(df.drop("PRICE", axis=1).shape))
    f.write("\n")

    f.write("Target shape   : {}".format(df["PRICE"].shape))
    f.write("\n")

    f.write("\nColumn names")
    f.write("\n")
```

```
    f.write(str(df.columns))
    f.write("\n")

    f.write("\nStatistical summary")
    f.write("\n")
    f.write(str(df.describe()))
    f.write("\n")

    f.write("\nDatatypes")
    f.write("\n")
    f.write(str(df.dtypes))
    f.write("\n")

    f.write("\nPEARSON correlation")
    f.write("\n")
    f.write(str(df.corr(method="pearson")))
    f.write("\n")

    f.write("\nSPEARMAN correlation")
    f.write("\n")
    f.write(str(df.corr(method="spearman")))
    f.write("\n")

    f.write("\nKENDALL correlation")
    f.write("\n")
    f.write(str(df.corr(method="kendall")))

    f.write("\nMissing Values")
    f.write("\n")
    f.write(str(pd.isnull(df).any()))
```

[ ]:

```
import random
import os

sns.set(color_codes=True)
colors = ["y", "b", "g", "r"]

cols = list(df.columns.values)

if not os.path.exists("plots/univariate/box"):
    os.makedirs("plots/univariate/box")

# draw a boxplot with vertical orientation
for i, col in enumerate(cols):
    sns.boxplot(df[col], color=random.choice(colors), orient="v")
    plt.savefig("plots/univariate/box/box_" + str(i) + ".png")
    plt.clf()
```

```python
    plt.close()
```

[23]:
```python
if not os.path.exists("plots/univariate/density"):
    os.makedirs("plots/univariate/density")

# draw a histogram and fit a kernel density estimate (KDE)
for i, col in enumerate(cols):
        sns.displot(df[col], color=random.choice(colors))
        plt.savefig("plots/univariate/density/density_" + str(i) + ".png")
        plt.clf()
        plt.close()
```

[25]:
```python
if not os.path.exists("plots/multivariate"):
    os.makedirs("plots/multivariate")

# bivariate plot between target and reason of absence
for i, col in enumerate(cols):
        if (i == len(cols) - 1):
                pass
        else:
                sns.jointplot(x=col, y="PRICE", data=df);
                plt.savefig("plots/multivariate/target_vs_" + str(i) + ".png")
                plt.clf()
                plt.close()
```

[26]:
```python
sns.pairplot(df)
plt.savefig("plots/pairplot.png")
plt.clf()
plt.close()
```

[27]:
```python
X = df.drop("PRICE", axis=1)
Y = df["PRICE"]
print(X.shape)
print(Y.shape)
```

```
(506, 13)
(506,)
```

[28]:
```python
from sklearn.preprocessing import StandardScaler, MinMaxScaler
scaler = MinMaxScaler().fit(X)
scaled_X = scaler.transform(X)
```

[29]:
```python
from sklearn.model_selection import train_test_split

seed      = 9
test_size = 0.20
```

```
X_train, X_test, Y_train, Y_test = train_test_split(scaled_X, Y, test_size =␣
↪test_size, random_state = seed)

print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

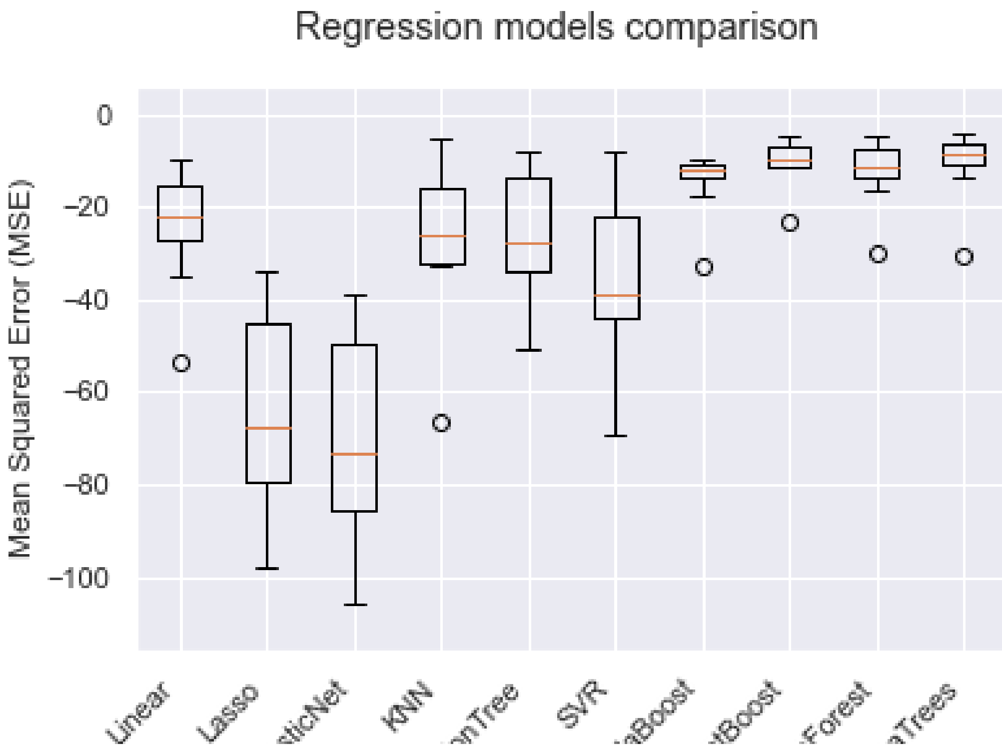
```
(404, 13)
(102, 13)
(404,)
(102,)
```

[31]:

```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.metrics import mean_squared_error

# user variables to tune
folds   = 10
metric  = "neg_mean_squared_error"

# hold different regression models in a single dictionary
models = {}
models["Linear"]        = LinearRegression()
models["Lasso"]         = Lasso()
models["ElasticNet"]    = ElasticNet()
models["KNN"]           = KNeighborsRegressor()
models["DecisionTree"]  = DecisionTreeRegressor()
models["SVR"]           = SVR()
models["AdaBoost"]      = AdaBoostRegressor()
models["GradientBoost"] = GradientBoostingRegressor()
models["RandomForest"]  = RandomForestRegressor()
models["ExtraTrees"]    = ExtraTreesRegressor()

# 10-fold cross validation for each model
model_results = []
model_names   = []
```

```python
for model_name in models:
        model   = models[model_name]
        k_fold  = KFold(n_splits=folds)
        results = cross_val_score(model, X_train, Y_train, cv=k_fold,␣
 ↪scoring=metric)

        model_results.append(results)
        model_names.append(model_name)
        print("{}: {}, {}".format(model_name, round(results.mean(), 3),␣
 ↪round(results.std(), 3)))

# box-whisker plot to compare regression models
figure = plt.figure()
figure.suptitle('Regression models comparison')
axis = figure.add_subplot(111)
plt.boxplot(model_results)
axis.set_xticklabels(model_names, rotation = 45, ha="right")
axis.set_ylabel("Mean Squared Error (MSE)")
plt.margins(0.05, 0.1)
plt.savefig("model_mse_scores.png")
plt.clf()
plt.close()
```

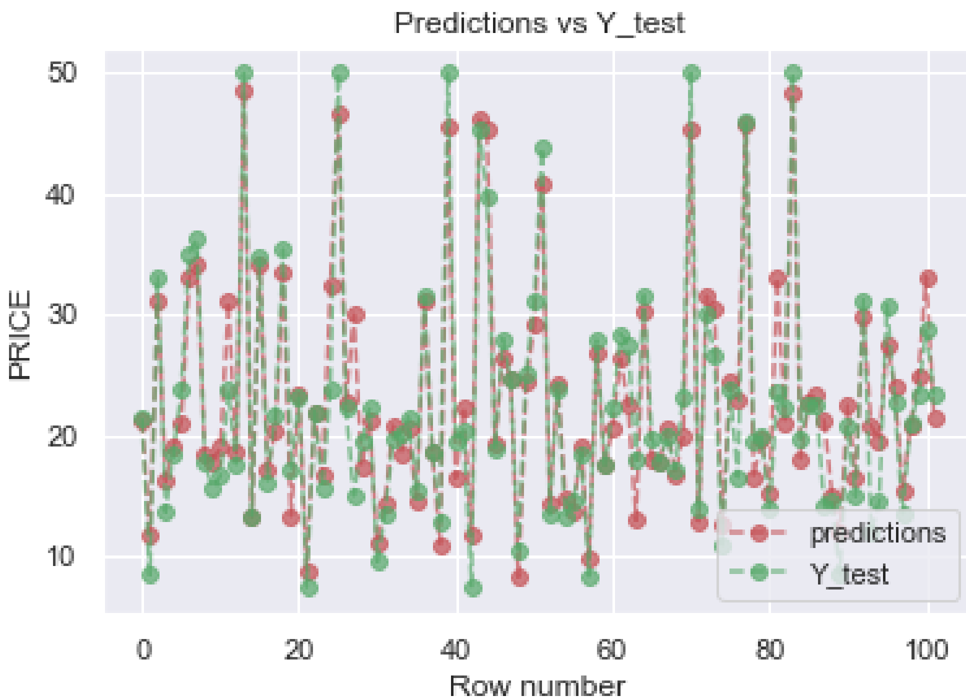
```
Linear: -23.794, 12.358
Lasso: -63.82, 20.646
ElasticNet: -69.362, 21.371
KNN: -26.366, 16.169
DecisionTree: -26.404, 13.326
SVR: -34.845, 17.408
AdaBoost: -14.375, 6.548
GradientBoost: -10.087, 5.029
RandomForest: -12.25, 6.972
ExtraTrees: -10.512, 7.225
```



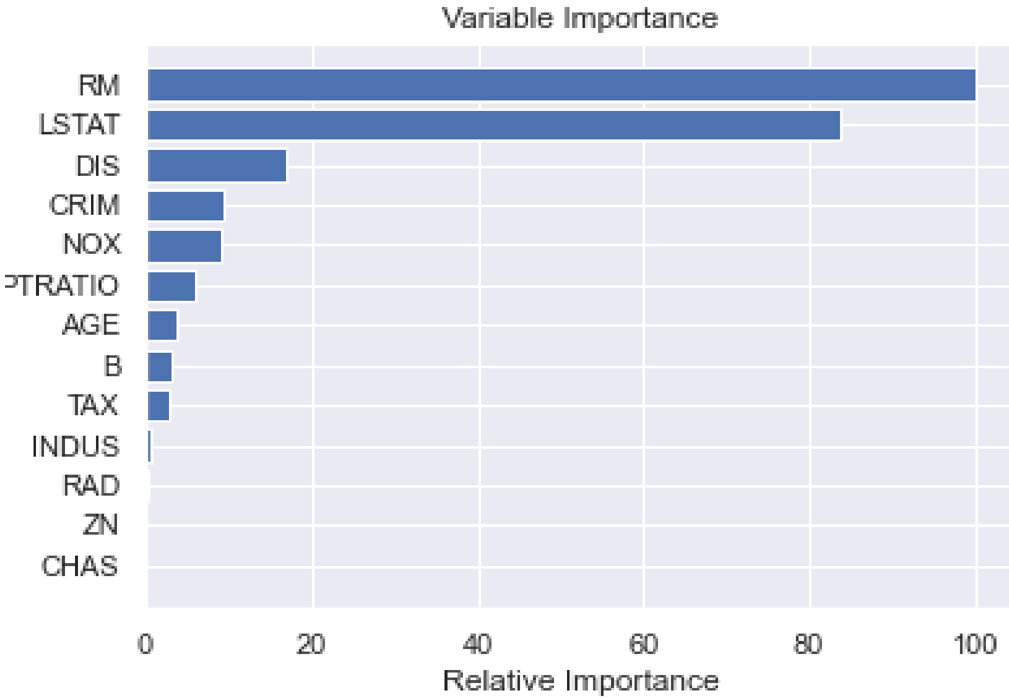
[32]:
```
# create and fit the best regression model
best_model = GradientBoostingRegressor(random_state=seed)
best_model.fit(X_train, Y_train)

# make predictions using the model
predictions = best_model.predict(X_test)
print("[INFO] MSE : {}".format(round(mean_squared_error(Y_test, predictions),␣
↪3)))
```

```
[INFO] MSE : 10.292
```

[36]:
```
# plot between predictions and Y_test
import numpy as np
x_axis = np.array(range(0, predictions.shape[0]))
plt.plot(x_axis, predictions, linestyle="--", marker="o", alpha=0.7, color='r',␣
↪label="predictions")
plt.plot(x_axis, Y_test, linestyle="--", marker="o", alpha=0.7, color='g',␣
↪label="Y_test")
plt.xlabel('Row number')
plt.ylabel('PRICE')
plt.title('Predictions vs Y_test')
plt.legend(loc='lower right')
plt.savefig("predictions_vs_ytest.png")
plt.clf()
plt.close()
```



[37]:
```
# plot model's feature importance
feature_importance = best_model.feature_importances_
feature_importance = 100.0 * (feature_importance / feature_importance.max())

sorted_idx = np.argsort(feature_importance)
pos        = np.arange(sorted_idx.shape[0]) + .5

plt.barh(pos, feature_importance[sorted_idx], align='center')
plt.yticks(pos, dataset.feature_names[sorted_idx])
plt.xlabel('Relative Importance')
plt.title('Variable Importance')
plt.savefig("feature_importance.png")
plt.clf()
plt.close()
```


Variable Importance
RM
LSTAT
DIS
CRIM
NOX
PTRATIO
AGE
B
TAX
INDUS
RAD
ZN
CHAS
0
20
40
60
80
100
Relative Importance

```
[ ]:
```