|  |  |  |
| --- | --- | --- |
| In [1]: | | *#Importing neccesary packages # Load libraries*  **from** sklearn.ensemble **import** GradientBoostingRegressor  **import** numpy **as** np  **import** pandas **as** pd  **from** sklearn.model\_selection **import** train\_test\_split  **from** sklearn.metrics **import** mean\_squared\_error  **from** sklearn.datasets **import** load\_boston  **from** sklearn.metrics **import** mean\_absolute\_error  **from** sklearn.metrics **import** r2\_score |
|  |  | **import** warnings |
|  |  | warnings**.**filterwarnings('ignore') |
|  |  |  |
| In | [2]: | *# Load data - Reading Boston Data* |
|  |  | boston **=** load\_boston() |
|  |  | X **=** pd**.**DataFrame(boston**.**data, columns**=**boston**.**feature\_names) *#Independent columns* |
|  |  | y **=** pd**.**Series(boston**.**target) *#Dependent column - Median value of House* |
|  |  |  |
| In | [3]: | *#Viewing Data - predictors*  X**.**head() |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Out[3]: | **CRIM** | **ZN** | **INDUS** | **CHAS** | **NOX** | **RM** | **AGE** | **DIS** | **RAD** | **TAX** | **PTRATIO** | **B** | **LSTAT** |
|  | **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 | 4.98 |
|  | **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 | 9.14 |
|  | **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 | 4.03 |
|  | **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 | 2.94 |
|  | **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 | 5.33 |

In [4]:

Out[4]:

In [5]:

1 21.6

y[1:10] *#response*

|  |  |
| --- | --- |
| 2 | 34.7 |
| 3 | 33.4 |
| 4 | 36.2 |
| 5 | 28.7 |
| 6 | 22.9 |
| 7 | 27.1 |
| 8 | 16.5 |
| 9 | 18.9 |

dtype: float64

*# Split dataset into training set and test set*

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X, y, test\_size**=**0.2) *# 80% tra*

In [6]:

*# Create gradientboost REGRESSOR object*

gradientregressor **=** GradientBoostingRegressor(max\_depth**=**2,n\_estimators**=**3,learning\_r

In [7]:

*# Train gradientboost REGRESSOR*

model **=** gradientregressor**.**fit(X\_train, y\_train)

*#Predict the response for test dataset*

y\_pred **=** model**.**predict(X\_test)

In [8]:

Out[8]:

In [9]:

**import** matplotlib.pyplot **as** plt

**%matplotlib** inline

*# Plot feature importance*

feature\_importance **=** model**.**feature\_importances\_

*# make importances relative to max importance*

feature\_importance **=** 100.0 **\*** (feature\_importance **/** feature\_importance**.**max()) sorted\_idx **=** np**.**argsort(feature\_importance)

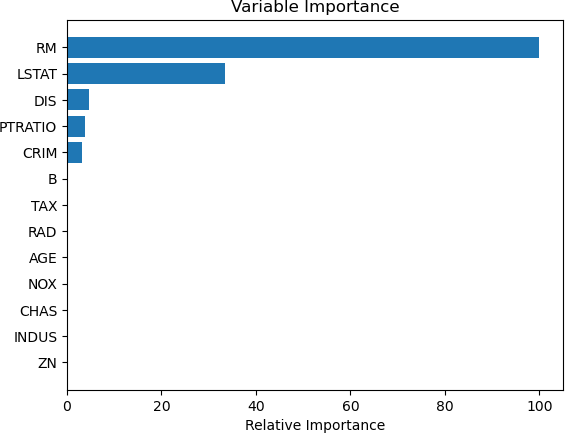
pos **=** np**.**arange(sorted\_idx**.**shape[0]) **+** .5

plt**.**barh(pos, feature\_importance[sorted\_idx], align**=**'center') plt**.**yticks(pos, boston**.**feature\_names[sorted\_idx])

plt**.**xlabel('Relative Importance') plt**.**title('Variable Importance') plt**.**show()

r2\_score(y\_pred,y\_test)

0.6594628107861282



In [11]:

**from** sklearn.model\_selection **import** GridSearchCV

LR **=** {'learning\_rate':[0.15,0.1,0.10,0.05], 'n\_estimators':[100,150,200,250]}

tuning **=** GridSearchCV(estimator **=**GradientBoostingRegressor(), param\_grid **=** LR, scoring**=**'r2')

tuning**.**fit(X\_train,y\_train)

tuning**.**best\_params\_, tuning**.**best\_score\_

Out[11]:

In [ ]:

({'learning\_rate': 0.1, 'n\_estimators': 250}, 0.8665225651189228)