In [1]:
```
import numpy as np
import pandas as pd
import scipy.stats as stats
import matplotlib.pyplot as plt
from sklearn.datasets import load_boston
import seaborn as sns
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
```

In [2]:
```
boston = load_boston()
bos = pd.DataFrame(boston.data, columns=boston.feature_names)
bos['MEDV'] = boston.target
```

In [3]:
```
bos.head()
```

Out[3]:

	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 [4]:
```
bos.isnull().sum()
```

Out[4]:
```
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
MEDV       0
dtype: int64
```

In [5]:
```
bos.describe()
```

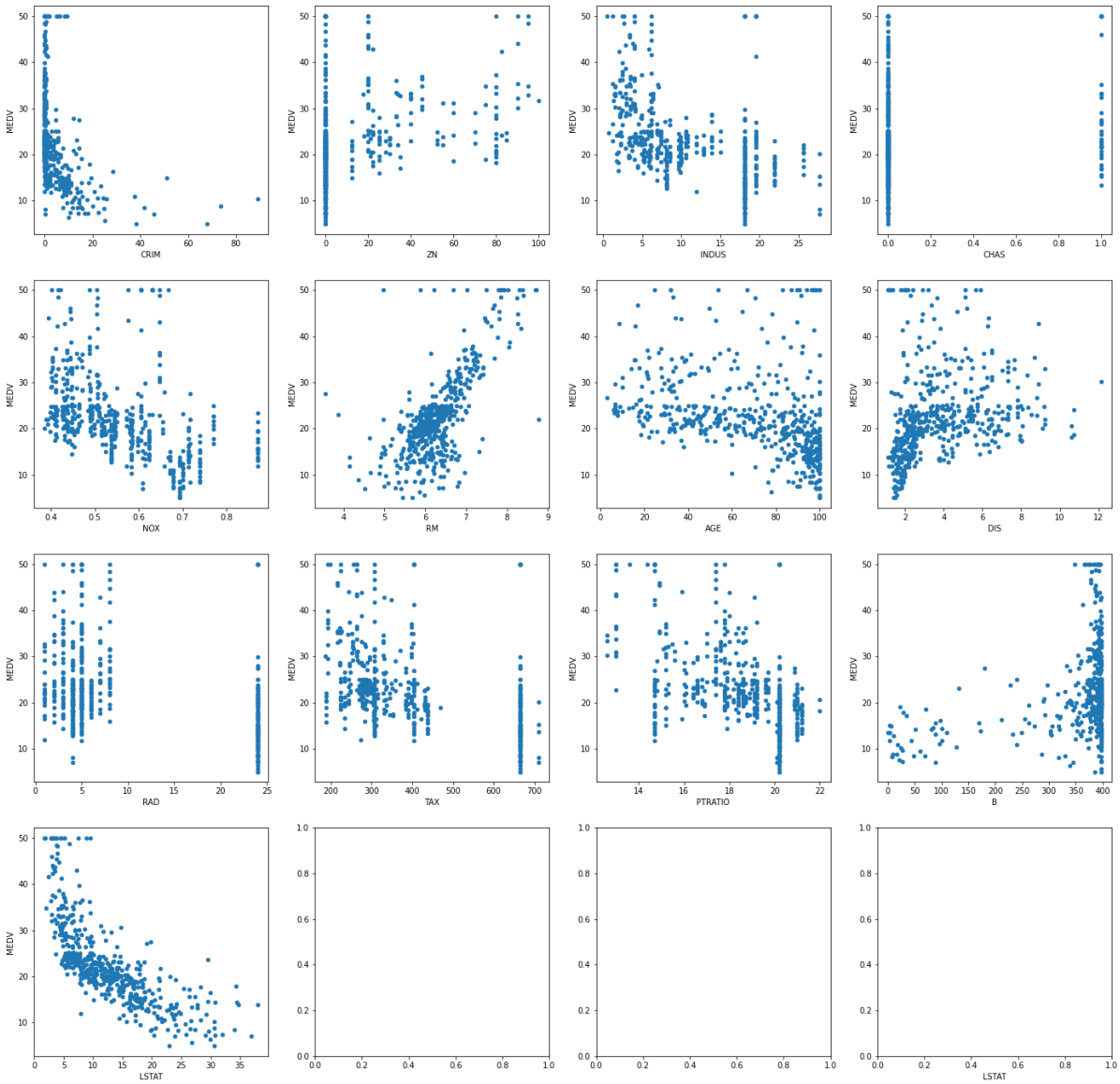
Out[5]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12

In [6]:
```
bos.info()
```

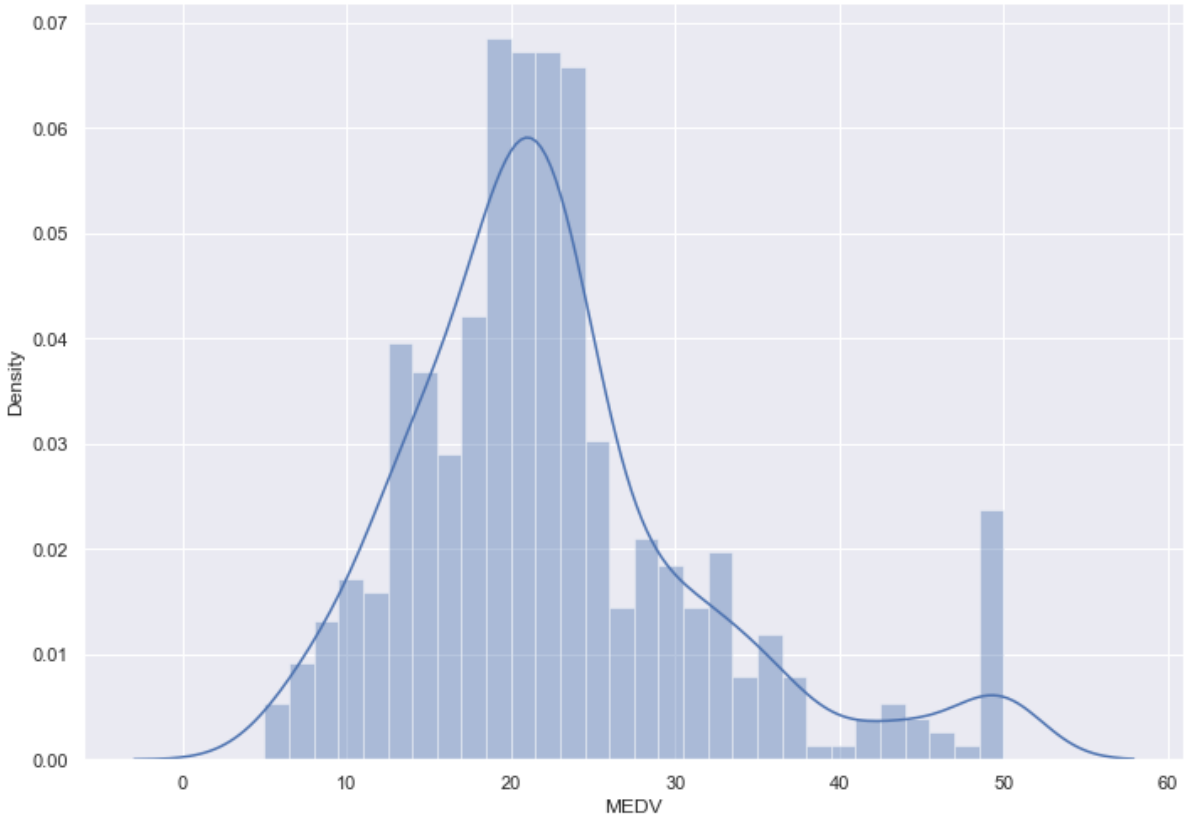
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    float64
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(14)
memory usage: 55.5 KB
```

In [7]:
```
# visualize the relationship between the features and the response using scatt
erplots
fig, axs = plt.subplots(4, 4, figsize=(24, 24))
# unpack all the axes subplots
axe = axs.ravel()
for i in range(len(bos.drop(columns='MEDV').columns)):
    bos.plot(kind='scatter', x=bos.columns[i], y='MEDV', ax=axe[i])
    plt.xlabel(bos.columns[i])
```



In [8]:
```
sns.set(rc={'figure.figsize':(11.7,8.27)})
sns.distplot(bos['MEDV'], bins=30)
plt.show()
```

```
C:\Users\Urvi\AppData\Roaming\Python\Python37\site-packages\seaborn\distribut
ions.py:2551: FutureWarning: `distplot` is a deprecated function and will be
removed in a future version. Please adapt your code to use either `displot`
(a figure-level function with similar flexibility) or `histplot` (an axes-lev
el function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [9]:
```
X = bos.drop(columns='MEDV')
y = bos.MEDV
```

Scaling data

In [10]:
```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

In [11]:
```
x_train, x_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3
, random_state=7)
```

In [12]:
```
rf = RandomForestRegressor()
rf.fit(x_train, y_train)
```

Out[12]:
```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=None, max_features='auto', max_leaf_nodes=Non
e,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=100, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)
```

In [13]:
```
rf.score(x_train, y_train)
```

Out[13]:
```
0.9764606744638792
```

In [14]:
```
rf.score(x_test, y_test)
```

Out[14]:
```
0.8154726481688918
```

Using grid search hyper parameter tuning

In [21]:
```
grid_param = {
    "n_estimators" : [90, 100],
    'criterion': ["mse", "mae"],
    'min_samples_leaf' : [1, 2, 3, 4, 5],
    'min_samples_split': [4, 5, 6, 7, 8],
    'max_features' : ['auto', 'log2']
}
```

In [22]:
```
grid_search = GridSearchCV(estimator=RandomForestRegressor(), param_grid=grid_
param, cv=5, n_jobs =-1, verbose = 3)
```

In [23]:
```
grid_search.fit(x_train, y_train)
```

```
Fitting 5 folds for each of 200 candidates, totalling 1000 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  16 tasks      | elapsed:    7.4s
[Parallel(n_jobs=-1)]: Done 112 tasks      | elapsed:   14.6s
[Parallel(n_jobs=-1)]: Done 272 tasks      | elapsed:   24.6s
[Parallel(n_jobs=-1)]: Done 496 tasks      | elapsed:   37.0s
[Parallel(n_jobs=-1)]: Done 784 tasks      | elapsed:  1.4min
[Parallel(n_jobs=-1)]: Done 1000 out of 1000 | elapsed:  1.6min finished
```

Out[23]:
```
GridSearchCV(cv=5, error_score=nan,
             estimator=RandomForestRegressor(bootstrap=True, ccp_alpha=0.0,
                                             criterion='mse', max_depth=None,
                                             max_features='auto',
                                             max_leaf_nodes=None,
                                             max_samples=None,
                                             min_impurity_decrease=0.0,
                                             min_impurity_split=None,
                                             min_samples_leaf=1,
                                             min_samples_split=2,
                                             min_weight_fraction_leaf=0.0,
                                             n_estimators=100, n_jobs=None,
                                             oob_score=False, random_state=No
ne,
                                             verbose=0, warm_start=False),
             iid='deprecated', n_jobs=-1,
             param_grid={'criterion': ['mse', 'mae'],
                         'max_features': ['auto', 'log2'],
                         'min_samples_leaf': [1, 2, 3, 4, 5],
                         'min_samples_split': [4, 5, 6, 7, 8],
                         'n_estimators': [90, 100]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=3)
```

In [24]:
```
grid_search.best_params_
```

Out[24]:
```
{'criterion': 'mae',
 'max_features': 'log2',
 'min_samples_leaf': 1,
 'min_samples_split': 6,
 'n_estimators': 90}
```

In [25]:
```
rf2 = RandomForestRegressor(criterion= 'mae', max_features = 'log2', min_sampl
es_leaf = 1, min_samples_split= 6,
                            n_estimators = 90, random_state=6)
```

In [26]:
```
rf2.fit(x_train,y_train)
```

Out[26]:
```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mae',
                      max_depth=None, max_features='log2', max_leaf_nodes=Non
e,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=6, min_weight_fraction_leaf=0.0,
                      n_estimators=90, n_jobs=None, oob_score=False,
                      random_state=6, verbose=0, warm_start=False)
```

In [28]:
```
rf2.score(x_train, y_train)
```

Out[28]:
```
0.9598513428646347
```

In [27]:
```
rf2.score(x_test, y_test)
```

Out[27]:
```
0.8173442972555908
```

Project Done By: Urvi Gadda

mailto: urvigada96@gmail.com

In [ ]:
```

```