```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

boston = pd.read_csv("/content/Boston.csv")

# Printing first 5 records of the dataset
print(dataset.head(5))
```

```
   Unnamed: 0     crim    zn  indus  chas    nox     rm   age     dis  rad  \
0           1  0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1
1           2  0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2
2           3  0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2
3           4  0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3
4           5  0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3

   tax  ptratio   black  lstat  medv
0  296     15.3  396.90   4.98  24.0
1  242     17.8  396.90   9.14  21.6
2  242     17.8  392.83   4.03  34.7
3  222     18.7  394.63   2.94  33.4
4  222     18.7  396.90   5.33  36.2
```

```
boston.shape
```

```
(506, 15)
```

```
boston.columns
```

```
Index(['Unnamed: 0', 'crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis',
       'rad', 'tax', 'ptratio', 'black', 'lstat', 'medv'],
      dtype='object')
```

```
data = pd.DataFrame(boston)
data.head(10)
```

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	bl
0	1	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396
1	2	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396
2	3	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392
3	4	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394
4	5	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396
5	6	0.02985	0.0	2.18	0	0.458	6.430	58.7	6.0622	3	222	18.7	394
6	7	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395
7	8	0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311	15.2	396
8	9	0.21124	12.5	7.87	0	0.524	5.631	100.0	6.0821	5	311	15.2	386

```
data.describe()
```

	Unnamed: 0	crim	zn	indus	chas	nox	rm
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	253.500000	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634
std	146.213884	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  506 non-null    int64  
 1   crim        506 non-null    float64
 2   zn          506 non-null    float64
 3   indus       506 non-null    float64
 4   chas        506 non-null    int64  
 5   nox         506 non-null    float64
 6   rm          506 non-null    float64
 7   age         506 non-null    float64
 8   dis         506 non-null    float64
 9   rad         506 non-null    int64  
 10  tax         506 non-null    int64  
 11  ptratio     506 non-null    float64
 12  black       506 non-null    float64
 13  lstat       506 non-null    float64
 14  medv        506 non-null    float64
dtypes: float64(11), int64(4)
memory usage: 59.4 KB
```

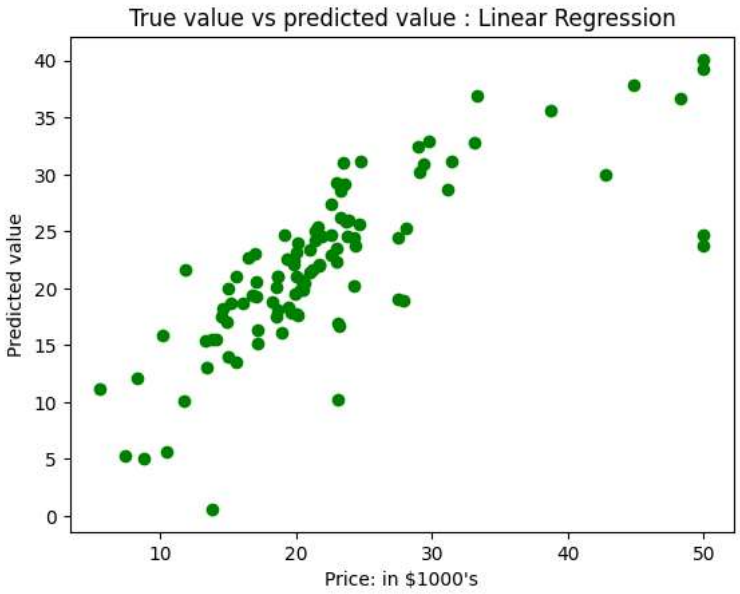
```
# Input Data
x = boston.drop('medv', axis=1)  # Assuming 'medv' is the target variable

# Output Data
y = boston['medv']
```

```
# splitting data to training and testing dataset.

#from sklearn.cross_validation import train_test_split
#the submodule cross_validation is renamed and deprecated to model_selection
from sklearn.model_selection import train_test_split

xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size =0.2,
                                                random_state = 0)

print("xtrain shape : ", xtrain.shape)
print("xtest shape : ", xtest.shape)
print("ytrain shape : ", ytrain.shape)
print("ytest shape : ", ytest.shape)
```

```
xtrain shape :  (404, 14)
xtest shape :  (102, 14)
ytrain shape :  (404,)
ytest shape :  (102,)
```

```
# Fitting Multi Linear regression model to training model
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(xtrain, ytrain)

# predicting the test set results
y_pred = regressor.predict(xtest)
```

```
# Plotting Scatter graph to show the prediction
# results - 'ytrue' value vs 'y_pred' value
plt.scatter(ytest, y_pred, c = 'green')
plt.xlabel("Price: in $1000's")
plt.ylabel("Predicted value")
plt.title("True value vs predicted value : Linear Regression")
plt.show()
```




```
from sklearn.metrics import mean_squared_error, mean_absolute_error
mse = mean_squared_error(ytest, y_pred)
mae = mean_absolute_error(ytest,y_pred)
print("Mean Square Error : ", mse)
print("Mean Absolute Error : ", mae)
```

```
Mean Square Error :  33.266961459239106
Mean Absolute Error :  3.838476893830883
```

```
from sklearn import svm
from sklearn.svm import SVC
from sklearn.metrics import mean_absolute_percentage_error

model_SVR = svm.SVR()
model_SVR.fit(X_train,Y_train)
Y_pred = model_SVR.predict(X_valid)

print(mean_absolute_percentage_error(Y_valid, Y_pred))
```

```
0.1870512931870423
```

```
from sklearn.ensemble import RandomForestRegressor

model_RFR = RandomForestRegressor(n_estimators=10)
model_RFR.fit(X_train, Y_train)
Y_pred = model_RFR.predict(X_valid)

mean_absolute_percentage_error(Y_valid, Y_pred)
```

```
0.1881558735534243
```

```
from sklearn.linear_model import LinearRegression

model_LR = LinearRegression()
model_LR.fit(X_train, Y_train)
Y_pred = model_LR.predict(X_valid)

print(mean_absolute_percentage_error(Y_valid, Y_pred))
```

```
0.18741683841599854
```

```
!pip3 install catboost
```

```
Requirement already satisfied: catboost in /usr/local/lib/python3.10/dist-packages (1.2.2)
Requirement already satisfied: graphviz in /usr/local/lib/python3.10/dist-packages (from catboost) (0.20.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from catboost) (3.7.1)
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.10/dist-packages (from catboost) (1.23.5)
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.10/dist-packages (from catboost) (1.5.3)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from catboost) (1.11.3)
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-packages (from catboost) (5.15.0)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from catboost) (1.16.0)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24->catboost) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24->catboost) (2023.3.post1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (1.1.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (4.43.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (23.2)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (3.1.1)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from plotly->catboost) (8.2.3)
```

```
# This code is contributed by @amartajisce
from catboost import CatBoostRegressor
cb_model = CatBoostRegressor()
cb_model.fit(X_train, Y_train)
preds = cb_model.predict(X_valid)
from sklearn.metrics import r2_score
cb_r2_score=r2_score(Y_valid, preds)
cb_r2_score
```

```
994:    learn: 24488.5023143    total: 1.91s    remaining: 9.62ms
995:    learn: 24468.9731701    total: 1.92s    remaining: 7.69ms
996:    learn: 24459.0497225    total: 1.92s    remaining: 5.77ms
997:    learn: 24458.7284279    total: 1.92s    remaining: 3.84ms
998:    learn: 24448.9981948    total: 1.92s    remaining: 1.92ms
999:    learn: 24440.7422679    total: 1.92s    remaining: 0us
0.38351169878113034
```

```
from sklearn.metrics import r2_score

# Calculate the R-squared score
r2 = r2_score(ytest, y_pred)

print("R-squared score:", r2)
```

```
R-squared score: 0.5914577039362054
```

```
"""As per the result, our model is only 66.55% accurate. So, the prepared model is not very good for predicting housing prices. One can improv

Here are a few further steps on how we can improve your model.

Feature Selection
Cross-Validation
Hyperparameter Tuning"""
```

```
'As per the result, our model is only 66.55% accurate. So, the prepared model is not ve
ry good for predicting housing prices. One can improve the prediction results using man
y other possible machine learning algorithms and techniques. \n\nHere are a few further
steps on how we can improve your model.\n\nFeature Selection\nCross-Validation\nHyperpa
```

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_regression

# Select the top k features based on their F-scores
k = 5  # You can choose the desired number of features
selector = SelectKBest(f_regression, k=k)
xtrain_selected = selector.fit_transform(xtrain, ytrain)
xtest_selected = selector.transform(xtest)
```

```
from sklearn.model_selection import cross_val_score

# Create and fit a linear regression model
regressor_cv = LinearRegression()

# Perform cross-validation and compute the R-squared score
cv_scores = cross_val_score(regressor_cv, xtrain_selected, ytrain, cv=5)
mean_cv_score = cv_scores.mean()
print("Cross-Validation R-squared Score: ", mean_cv_score)
```

```
Cross-Validation R-squared Score:  0.7153642067564461
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_regression
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV

# Standardize your features
scaler = StandardScaler()
xtrain_scaled = scaler.fit_transform(xtrain)
xtest_scaled = scaler.transform(xtest)

# Feature selection
k = 5  # You can choose the desired number of features
selector = SelectKBest(f_regression, k=k)
xtrain_selected = selector.fit_transform(xtrain_scaled, ytrain)
xtest_selected = selector.transform(xtest_scaled)

# Create and fit a linear regression model
regressor_cv = LinearRegression()

# Perform cross-validation and compute the R-squared score
```

```
cv_scores = cross_val_score(regressor_cv, xtrain_selected, ytrain, cv=5)
mean_cv_score = cv_scores.mean()
print("Cross-Validation R-squared Score: ", mean_cv_score)

# Hyperparameter Tuning (if you are using Ridge or Lasso)
param_grid = {'alpha': [0.001, 0.01, 0.1, 1, 10, 100]}
regressor_tuned = GridSearchCV(Ridge(), param_grid, cv=5)
regressor_tuned.fit(xtrain_selected, ytrain)
best_params = regressor_tuned.best_params_
print("Best Hyperparameters: ", best_params)
```

```
Cross-Validation R-squared Score:  0.7153642067564474
Best Hyperparameters:  {'alpha': 10}
```