```
#boston house price
```

```
import pandas as pd
import numpy as np
from sklearn import metrics
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
# This cell was incomplete and is no longer needed.
```

```
from google.colab import files
uploaded = files.upload()
```

```
Choose files No file chosen     Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving BostonHousing.csv to BostonHousing.csv
```

```
data=pd.read_csv('BostonHousing.csv')
```

```
data.head()
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat	medv
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

```
data.shape
```

```
(506, 14)
```

```
data.column()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/tmp/ipython-input-3605313122.py in <cell line: 0>()
----> 1 data.column()

/usr/local/lib/python3.12/dist-packages/pandas/core/generic.py in __getattr__(self, name)
   6297         ):
   6298             return self[name]
-> 6299         return object.__getattribute__(self, name)
   6300 
   6301     @final

AttributeError: 'DataFrame' object has no attribute 'column'
```

```
data.dtypes
```

	0
crim	float64
zn	float64
indus	float64
chas	int64
nox	float64
rm	float64
age	float64
dis	float64
rad	int64
tax	int64
ptratio	float64
b	float64
lstat	float64
medv	float64

dtype: object

```
#identifying the unique
```

```
data.nunique()
```

```
          0
 crim   504
 zn      26
 indus   76
 chas     2
 nox     81
 rm     441
 age    356
 dis    412
 rad      9
 tax     66
 ptratio 46
 b      357
 lstat  455
 medv   229

dtype: int64
```

```
data.describe()
```

```
data.isnull
```

```
pandas.core.frame.DataFrame.isnull
def isnull() -> DataFrame

DataFrame.isnull is an alias for DataFrame.isna.

Detect missing values.

Return a boolean same-sized object indicating if the values are NA.
NA values, such as None or :attr:`numpy.NaN`, gets mapped to True
```

```
# check for missing
data.isnull().sum()
```

```
          0
 crim     0
 zn       0
 indus    0
 chas     0
 nox      0
 rm       5
 age      0
 dis      0
 rad      0
 tax      0
 ptratio  0
 b        0
 lstat    0
 medv     0

dtype: int64
```

```
Start coding or generate with AI.
```

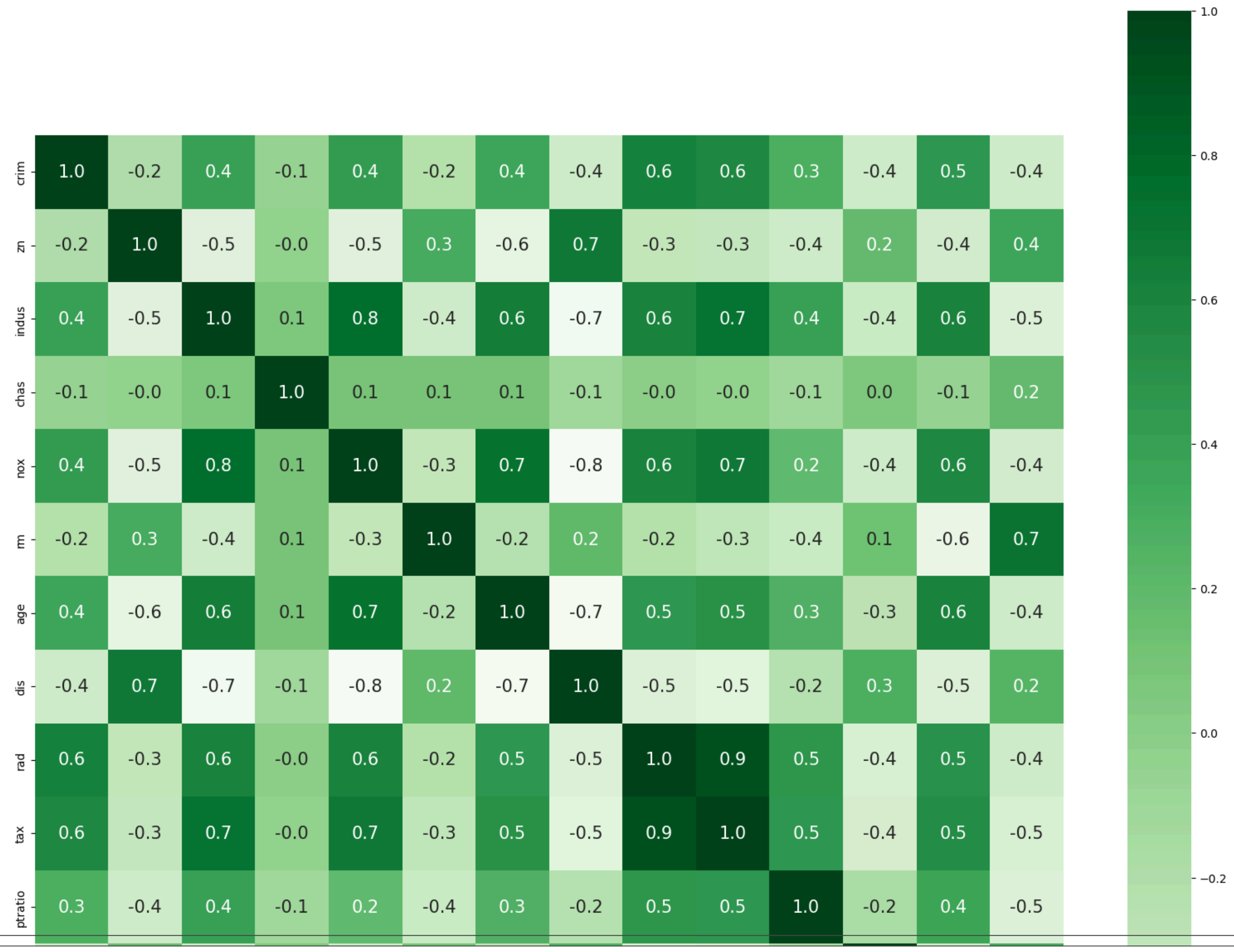
```
Start coding or generate with AI.
```

```
corr = data.corr()
corr.shape
```

```
(14, 14)
```

```
#plotting the heatmap of correlation between feature
plt.figure(figsize=(20,20))
sns.heatmap(corr, cbar=True, square = True, fmt='.1f', annot=True, annot_kws={'size': 15}, cmap= 'Greens')
#
```

```
<Axes: >
```



```
# spliting target variable and independent variable
x=data.drop('medv',axis=1)
y=data['medv']
```

```
# Splitting

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=4)
```

```
# import library for Linear regression
from sklearn.linear_model import LinearRegression

# Impute missing values in 'rm' column with the mean
x_train['rm'] = x_train['rm'].fillna(x_train['rm'].mean())
x_test['rm'] = x_test['rm'].fillna(x_test['rm'].mean())

#Create a Linear
lm=LinearRegression()
#Train the model using the training sets
lm.fit(x_train,y_train)
```

```
/tmp/ipython-input-269149820.py:5: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform


  x_train['rm'].fillna(x_train['rm'].mean(), inplace=True)
/tmp/ipython-input-269149820.py:6: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform


  x_test['rm'].fillna(x_test['rm'].mean(), inplace=True)
```

```
▾ LinearRegression
LinearRegression()
```

```
#   Value of y intercept
lm.intercept_
```

```
np.float64(36.43173988495616)
```

```
#converting the coefficent value to a dataframe
coefficients = pd.DataFrame(lm.coef_, x_train.columns, columns=['Coefficient'])
coefficients
```

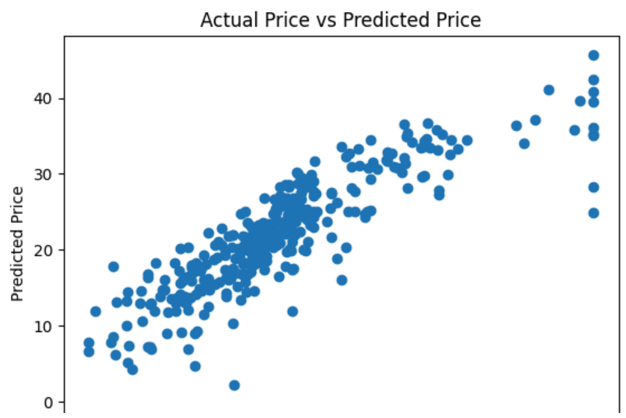
	Coefficient
crim	-0.122574
zn	0.055749
indus	-0.007747
chas	4.698119
nox	-14.435639
rm	3.268024
age	-0.003222
dis	-1.547625
rad	0.326319
tax	-0.014067
ptratio	-0.805640
b	0.009357
lstat	-0.523870

```
#model evaluation
y_pred=lm.predict(x_train)
```

```
# model evalution
print('R^2:',metrics.r2_score(y_train,y_pred))
print('Adjusted R^2:', 1 - (1 - metrics.r2_score(y_train, y_pred)) * (len(y_train) - 1) / (len(y_train) - x_train.shape[1] - 1))
print('MAE:', metrics.mean_absolute_error(y_train,y_pred))
print('MSE:', metrics.mean_squared_error(y_train, y_pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_train,y_pred)))
```

```
R^2: 0.7461675856234777
Adjusted R^2: 0.736462228603199
MAE: 3.092425645890494
MSE: 19.106176425384987
RMSE: 4.3710612470411565
```

```
plt.scatter(y_train,y_pred)
plt.xlabel("Price")
plt.ylabel("Predicted Price")
plt.title("Actual Price vs Predicted Price")
plt.show()
```


Actual Price vs Predicted Price
Predicted Price