**import** **numpy** **as** **np**

**import** **pandas** **as** **pd**

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

**import** **seaborn** **as** **sns**

**import** **sklearn.datasets**

**from** **sklearn.model\_selection** **import** train\_test\_split

**from** **xgboost** **import** XGBRegressor

**from** **sklearn** **import** metrics

In [6]:

house\_price\_dataset = sklearn.datasets.load\_boston()

In [ ]:

print(house\_price\_dataset)

In [ ]:

house\_price\_df = pd.DataFrame(house\_price\_dataset.data)

print(house\_price\_df)

In [ ]:

house\_price\_df = pd.DataFrame(house\_price\_dataset.data, columns= house\_price\_dataset.feature\_names)

print(house\_price\_df)

In [ ]:

house\_price\_df["Price"]= house\_price\_dataset.target

print(house\_price\_df)

In [20]:

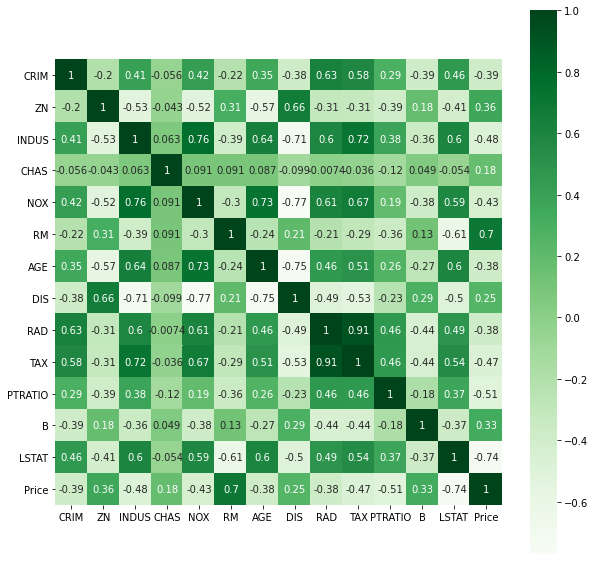
correaltion = house\_price\_df.corr()

plt.figure(figsize=(10,10))

sns.heatmap(correaltion, cbar=**True**, square=**True**, annot=**True**, annot\_kws={'size':10}, cmap="Greens")

Out[20]:

<AxesSubplot:>



In [ ]:

In [14]:

x = house\_price\_df.drop(["Price"], axis=1)

y = house\_price\_df["Price"]

In [ ]:

print(x,y)

In [22]:

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size=0.20, random\_state=2)

print(x.shape,x\_train.shape,x\_test.shape, y\_train.shape,y\_test.shape)

(506, 13) (404, 13) (102, 13) (404,) (102,)

In [33]:

model1 = XGBRegressor()

In [36]:

model1.fit(x\_train, y\_train)

Out[36]:

XGBRegressor(base\_score=0.5, booster='gbtree', colsample\_bylevel=1,

colsample\_bynode=1, colsample\_bytree=1, gamma=0, gpu\_id=-1,

importance\_type='gain', interaction\_constraints='',

learning\_rate=0.300000012, max\_delta\_step=0, max\_depth=6,

min\_child\_weight=1, missing=nan, monotone\_constraints='()',

n\_estimators=100, n\_jobs=4, num\_parallel\_tree=1, random\_state=0,

reg\_alpha=0, reg\_lambda=1, scale\_pos\_weight=1, subsample=1,

tree\_method='exact', validate\_parameters=1, verbosity=None)

In [ ]:

training\_data\_prediction = model1.predict(x\_train)

print(training\_data\_prediction)

In [39]:

score\_1 = metrics.r2\_score(y\_train, training\_data\_prediction)

score\_2 = metrics.mean\_absolute\_error(y\_train, training\_data\_prediction)

print(score\_1,score\_2)

0.9999948236320982 0.0145848437110976

In [ ]:

test\_data\_prediction = model1.predict(x\_test)

print(test\_data\_prediction)

In [49]:

score\_3 = metrics.r2\_score(y\_test, test\_data\_prediction)

score\_4 = metrics.mean\_absolute\_error(y\_test, test\_data\_prediction)

print(score\_3,score\_4)

0.8711660369151691 2.2834744154238233

In [53]:

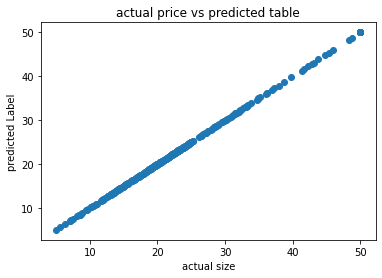
plt.scatter(y\_train, training\_data\_prediction)

plt.xlabel("actual size")

plt.ylabel("predicted Label")

plt.title("actual price vs predicted table")

plt.show()



In [ ]: