**Algorithm**

**(Code)**

1. **Linear regression:**

* import pandas as pd

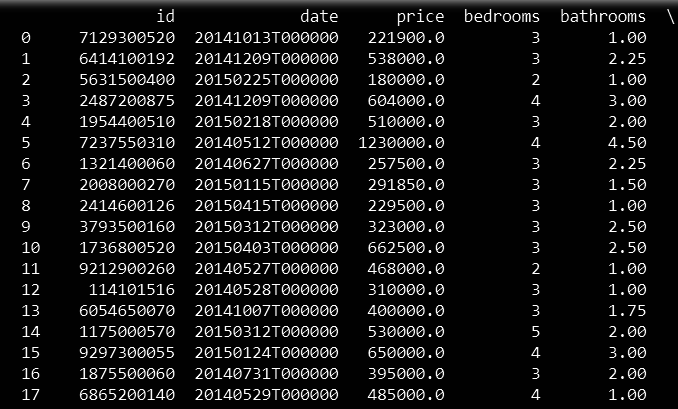
import numpy as np

import matplotlib.pyplot as plt

import seaborn as snc

* data = pd.read\_csv("housingnew.csv")

print(data)



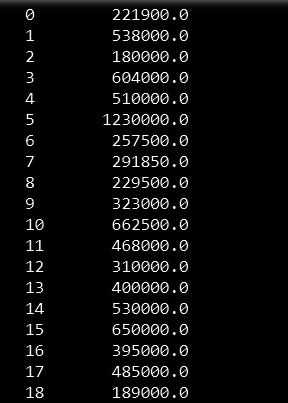
* input = data.drop(["id","date","price"],axis = 1)

outcome = data.price

print(input)



* print(outcome)



* from sklearn.linear\_model import LinearRegression

linreg = LinearRegression()

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

x\_train,x\_test,y\_train,y\_test = train\_test\_split(input,outcome,test\_size = 0.2,random\_state=2)

* linreg.fit(x\_train,y\_train)



* linreg.score(x\_test,y\_test)



* plt.scatter(linreg.predict(input), linreg.predict(input) - outcome,

color = "green", s = 10, label = 'Train data')

***## plotting residual errors in test data***

plt.scatter(linreg.predict(x\_test), linreg.predict(x\_test) - y\_test,

color = "blue", s = 10, label = 'Test data')

***## plotting line for zero residual error***

plt.hlines(y = 0, xmin = -500000, xmax = 3500000, linewidth = 2)

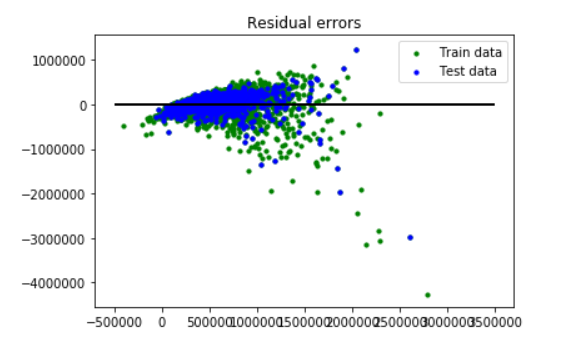
plt.legend(loc = 'upper right')

***## plot title***

plt.title("Residual errors")

***## function to show plot***

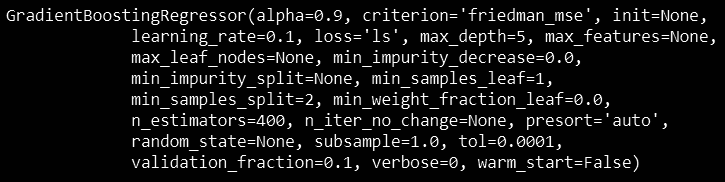
plt.show()



* from sklearn import ensemble

clf = ensemble.GradientBoostingRegressor(n\_estimators = 400, max\_depth = 5, min\_samples\_split = 2,learning\_rate = 0.1, loss = 'ls')

* clf.fit(x\_train, y\_train)



* clf.score(x\_test,y\_test)



* plt.scatter(clf.predict(input), clf.predict(input) - outcome, color = "green", s = 10, label = 'Train data')

***## plotting residual errors in test data***

plt.scatter(clf.predict(x\_test), clf.predict(x\_test) - y\_test, color = "blue", s = 10,label = 'Test data')

***## plotting line for zero residual error***

plt.hlines(y = 0, xmin = 0, xmax = 3500000, linewidth = 2)

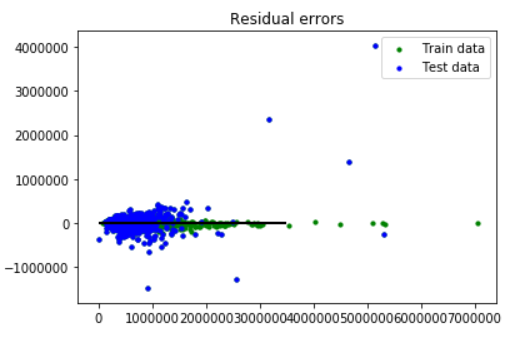
plt.legend(loc = 'upper right')

***## plot title***

plt.title("Residual errors")

***## function to show plot***

plt.show()

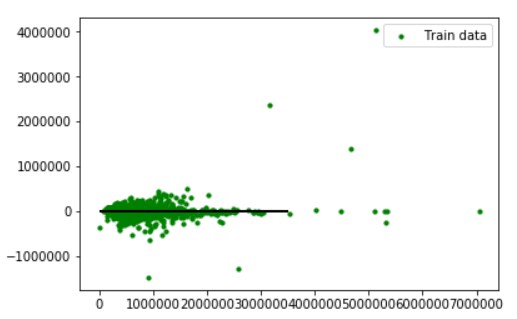


* plt.scatter(clf.predict(input), clf.predict(input) - outcome,color = "green", s = 10, label = 'Train data')

plt.hlines(y = 0, xmin = 0, xmax = 3500000, linewidth = 2)

plt.legend(loc = 'upper right')

plt.show()



* plt.scatter(clf.predict(x\_test), clf.predict(x\_test) - y\_test,color = "blue", s = 10, label = 'Test data')

plt.hlines(y = 0, xmin = 0, xmax = 3500000, linewidth = 2)

plt.legend(loc = 'upper right')

plt.show()

