**Algorithm**

**(Code)**

**3) Random Forest Tree:**

* import numpy as np

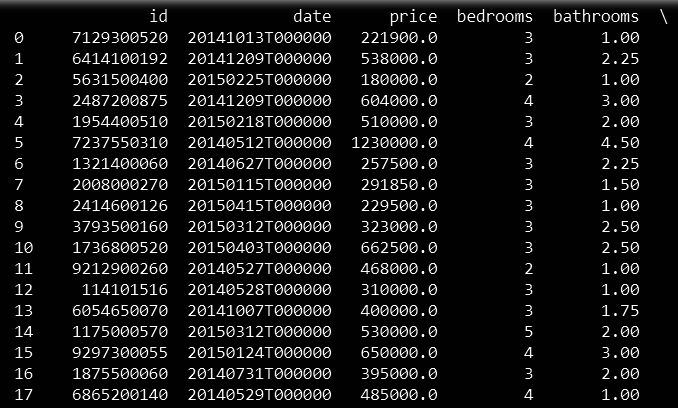
import pandas as pd

%matplotlib inline

import matplotlib.pyplot as plt

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

print(data)



* data.describe()

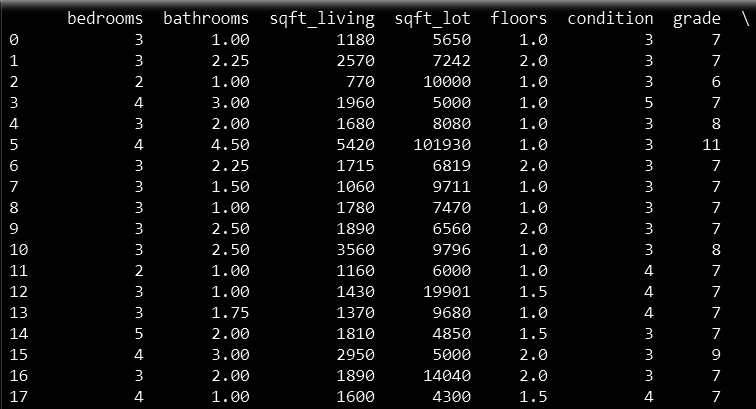


* X = data.drop(["id","date","price"], axis = 1)

y = data.price

print(X)

print(y)



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

x\_training\_set, x\_test\_set, y\_training\_set, y\_test\_set = train\_test\_split(X,y,test\_size=0.20, random\_state=100,shuffle=True)

* from sklearn.ensemble import RandomForestRegressor

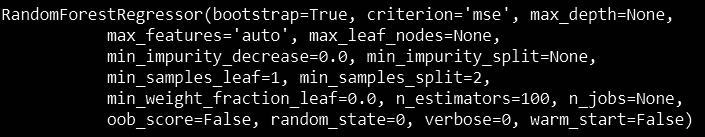
n\_estimators=100

***# Fit regression model***

***# Estimate the score on the entire dataset, with no missing values***

model = RandomForestRegressor(random\_state=0, n\_estimators=n\_estimators)

model.fit(x\_training\_set, y\_training\_set)



* model.score(x\_test\_set,y\_test\_set)



* from sklearn.metrics import mean\_squared\_error, r2\_score

model\_score = model.score(x\_training\_set,y\_training\_set)

***# Have a look at R sq to give an idea of the fit ,***

***# Explained variance score: 1 is perfect prediction***

y\_predicted = model.predict(x\_test\_set)

***# The mean squared error***

print("Mean squared error: %.2f"% mean\_squared\_error(y\_test\_set, y\_predicted))

***# Explained variance score: 1 is perfect prediction***

print('Test Variance score: %.2f' % r2\_score(y\_test\_set, y\_predicted))



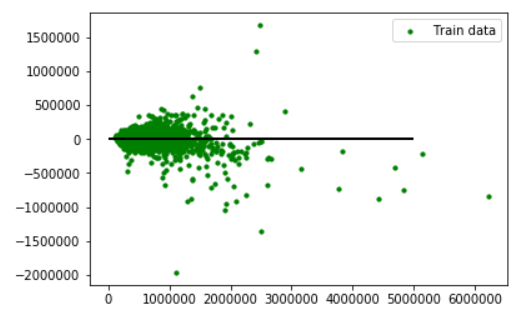
* plt.scatter(model.predict(X), model.predict(X) - y,

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

plt.hlines(y = 0, xmin = 0, xmax = 5000000, linewidth = 2)

plt.legend(loc = 'upper right')

plt.show()



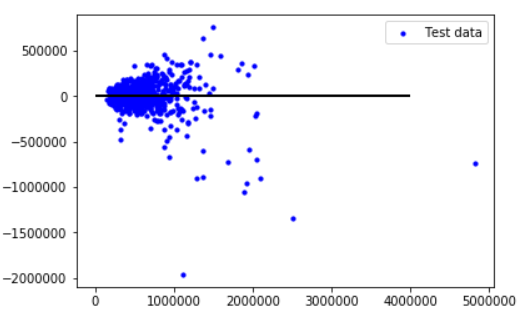
* plt.scatter(model.predict(x\_test\_set), model.predict(x\_test\_set) - y\_test\_set,

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

plt.hlines(y = 0, xmin = 0, xmax = 4000000, linewidth = 2)

plt.legend(loc = 'upper right')

plt.show()



* plt.scatter(model.predict(X), model.predict(X) - y,

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

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

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

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

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

plt.hlines(y = 0, xmin = 0, xmax = 5000000, linewidth = 2)

plt.legend(loc = 'upper right')

***## plot title***

plt.title("Residual errors")

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

plt.show()

