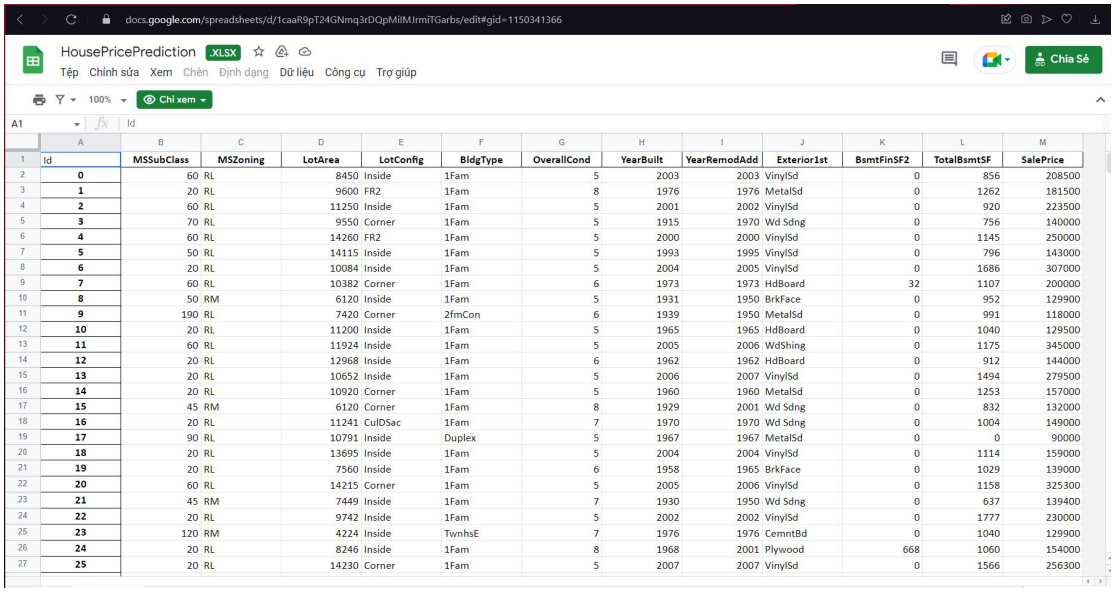
Dự án : tạo AI

Thành viên nhóm :
- Đặng Hoàng Phúc - 21022071
- Bùi Trọng Nghĩa - 21004501
- Lê Quang Vũ

Lấy database từ web về để làm data , tạo AI để dự đoán giá nhà

- Import các thư viện cần thiết pandas , numpy matplotlib.pyplot
- Lấy data từ file csv đã tải về mang tên : kc_house_data1.csv



docs.google.com/spreadsheets/d/1caaR9pT24GNmq3rDQpMilMJrmiTGarbs/edit#gid=1150341366

HousePricePrediction .XLSX

Tệp Chỉnh sửa Xem Chèn Định dạng Dữ liệu Công cụ Trợ giúp

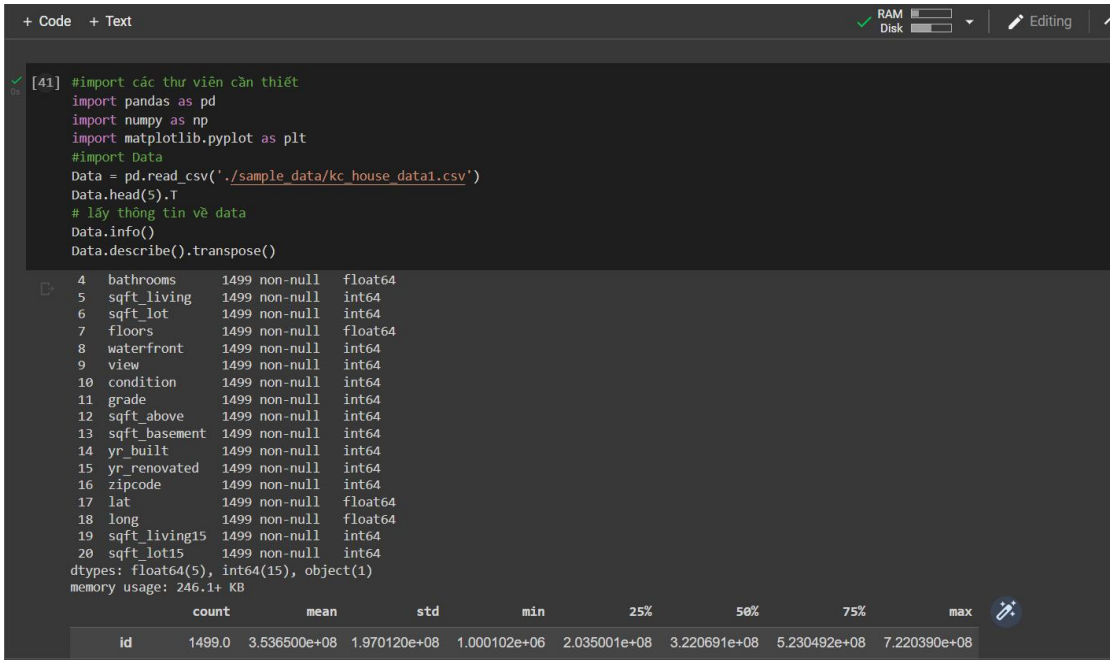
Chia Sẻ

100% Chỉ xem

A1 fx Id

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Id	MSSubClass	MSZoning	LotArea	LotConfig	BldgType	OverallCond	YearBuilt	YearRemodAdd	Exterior1st	BsmtFinSF2	TotalBsmtSF	SalePrice
2	0	60	RL	8450	Inside	1Fam	5	2003	2003	VinylSd	0	856	208500
3	1	20	RL	9600	FR2	1Fam	8	1976	1976	MetalSd	0	1262	181500
4	2	60	RL	11250	Inside	1Fam	5	2001	2002	VinylSd	0	920	223500
5	3	70	RL	9550	Corner	1Fam	5	1915	1970	Wd Sdng	0	756	140000
6	4	60	RL	14260	FR2	1Fam	5	2000	2000	VinylSd	0	1145	250000
7	5	50	RL	14115	Inside	1Fam	5	1993	1995	VinylSd	0	796	143000
8	6	20	RL	10084	Inside	1Fam	5	2004	2005	VinylSd	0	1686	307000
9	7	60	RL	10382	Corner	1Fam	6	1973	1973	HdBoard	32	1107	200000
10	8	50	RM	6120	Inside	1Fam	5	1931	1950	BrkFace	0	952	129900
11	9	190	RL	7420	Corner	2fmCon	6	1939	1950	MetalSd	0	991	118000
12	10	20	RL	11200	Inside	1Fam	5	1965	1965	HdBoard	0	1040	129500
13	11	60	RL	11924	Inside	1Fam	5	2005	2006	WdShing	0	1175	345000
14	12	20	RL	12968	Inside	1Fam	6	1962	1962	HdBoard	0	912	144000
15	13	20	RL	10652	Inside	1Fam	5	2006	2007	VinylSd	0	1494	279500
16	14	20	RL	10920	Corner	1Fam	5	1960	1960	MetalSd	0	1253	157000
17	15	45	RM	6120	Corner	1Fam	8	1929	2001	Wd Sdng	0	832	132000
18	16	20	RL	11241	CulDSac	1Fam	7	1970	1970	Wd Sdng	0	1004	149000
19	17	90	RL	10791	Inside	Duplex	5	1967	1967	MetalSd	0	0	90000
20	18	20	RL	13695	Inside	1Fam	5	2004	2004	VinylSd	0	1114	159000
21	19	20	RL	7560	Inside	1Fam	6	1958	1965	BrkFace	0	1029	139000
22	20	60	RL	14215	Corner	1Fam	5	2005	2006	VinylSd	0	1158	325300
23	21	45	RM	7449	Inside	1Fam	7	1930	1950	Wd Sdng	0	637	139400
24	22	20	RL	9742	Inside	1Fam	5	2002	2002	VinylSd	0	1777	230000
25	23	120	RM	4224	Inside	TwnhsE	7	1976	1976	CemntBd	0	1040	129900
26	24	20	RL	8246	Inside	1Fam	8	1968	2001	Plywood	668	1060	154000
27	25	20	RL	14230	Corner	1Fam	5	2007	2007	VinylSd	0	1566	256300

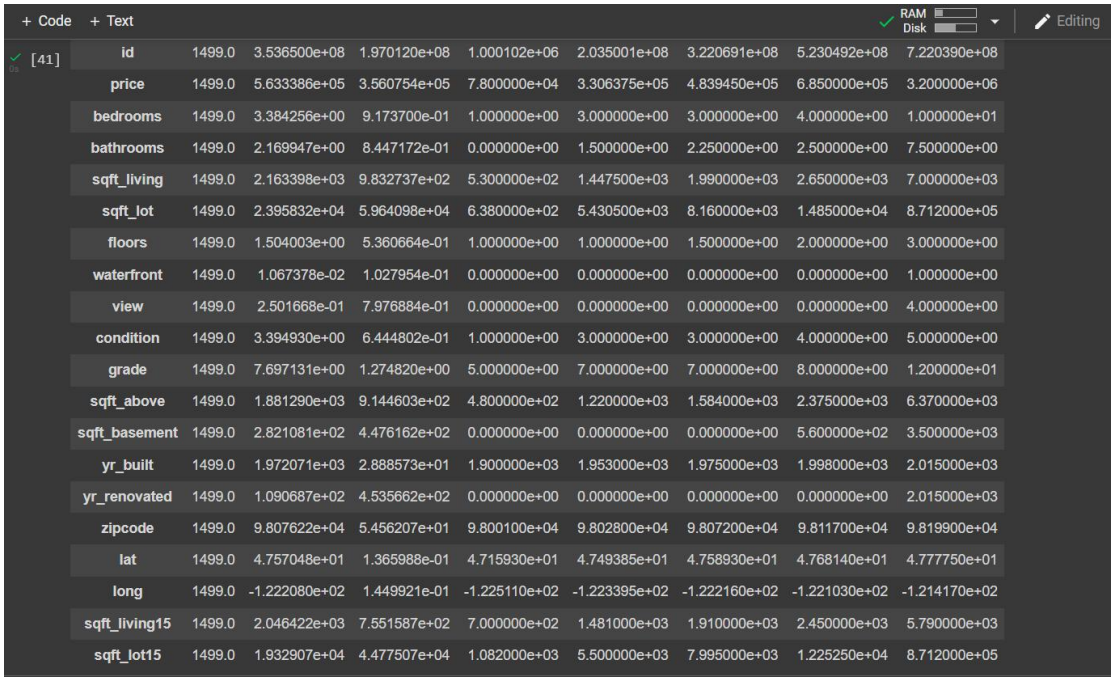
- In ra thông tin của data : Data.info()
- Mô tả data và đổi chỗ cho data : Data.describe().transpose()



```
[41] #import các thư viện cần thiết
     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     #import Data
     Data = pd.read_csv('./sample_data/kc_house_data1.csv')
     Data.head(5).T
     # lấy thông tin về data
     Data.info()
     Data.describe().transpose()
```

```
 4   bathrooms      1499 non-null   float64
 5   sqft_living    1499 non-null   int64
 6   sqft_lot       1499 non-null   int64
 7   floors         1499 non-null   float64
 8   waterfront     1499 non-null   int64
 9   view           1499 non-null   int64
 10  condition      1499 non-null   int64
 11  grade          1499 non-null   int64
 12  sqft_above     1499 non-null   int64
 13  sqft_basement  1499 non-null   int64
 14  yr_built       1499 non-null   int64
 15  yr_renovated   1499 non-null   int64
 16  zipcode        1499 non-null   int64
 17  lat            1499 non-null   float64
 18  long           1499 non-null   float64
 19  sqft_living15  1499 non-null   int64
 20  sqft_lot15     1499 non-null   int64
dtypes: float64(5), int64(15), object(1)
memory usage: 246.1+ KB
```

	count	mean	std	min	25%	50%	75%	max
id	1499.0	3.536500e+08	1.970120e+08	1.000102e+06	2.035001e+08	3.220691e+08	5.230492e+08	7.220390e+08



[41]

id	1499.0	3.536500e+08	1.970120e+08	1.000102e+06	2.035001e+08	3.220691e+08	5.230492e+08	7.220390e+08
price	1499.0	5.633386e+05	3.560754e+05	7.800000e+04	3.306375e+05	4.839450e+05	6.850000e+05	3.200000e+06
bedrooms	1499.0	3.384256e+00	9.173700e-01	1.000000e+00	3.000000e+00	3.000000e+00	4.000000e+00	1.000000e+01
bathrooms	1499.0	2.169947e+00	8.447172e-01	0.000000e+00	1.500000e+00	2.250000e+00	2.500000e+00	7.500000e+00
sqft_living	1499.0	2.163398e+03	9.832737e+02	5.300000e+02	1.447500e+03	1.990000e+03	2.650000e+03	7.000000e+03
sqft_lot	1499.0	2.395832e+04	5.964098e+04	6.380000e+02	5.430500e+03	8.160000e+03	1.485000e+04	8.712000e+05
floors	1499.0	1.504003e+00	5.360664e-01	1.000000e+00	1.000000e+00	1.500000e+00	2.000000e+00	3.000000e+00
waterfront	1499.0	1.067378e-02	1.027954e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.000000e+00
view	1499.0	2.501668e-01	7.976884e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	4.000000e+00
condition	1499.0	3.394930e+00	6.444802e-01	1.000000e+00	3.000000e+00	3.000000e+00	4.000000e+00	5.000000e+00
grade	1499.0	7.697131e+00	1.274820e+00	5.000000e+00	7.000000e+00	7.000000e+00	8.000000e+00	1.200000e+01
sqft_above	1499.0	1.881290e+03	9.144603e+02	4.800000e+02	1.220000e+03	1.584000e+03	2.375000e+03	6.370000e+03
sqft_basement	1499.0	2.821081e+02	4.476162e+02	0.000000e+00	0.000000e+00	0.000000e+00	5.600000e+02	3.500000e+03
yr_built	1499.0	1.972071e+03	2.888573e+01	1.900000e+03	1.953000e+03	1.975000e+03	1.998000e+03	2.015000e+03
yr_renovated	1499.0	1.090687e+02	4.535662e+02	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	2.015000e+03
zipcode	1499.0	9.807622e+04	5.456207e+01	9.800100e+04	9.802800e+04	9.807200e+04	9.811700e+04	9.819900e+04
lat	1499.0	4.757048e+01	1.365988e-01	4.715930e+01	4.749385e+01	4.758930e+01	4.768140e+01	4.777750e+01
long	1499.0	-1.222080e+02	1.449921e-01	-1.225110e+02	-1.223395e+02	-1.222160e+02	-1.221030e+02	-1.214170e+02
sqft_living15	1499.0	2.046422e+03	7.551587e+02	7.000000e+02	1.481000e+03	1.910000e+03	2.450000e+03	5.790000e+03
sqft_lot15	1499.0	1.932907e+04	4.477507e+04	1.082000e+03	5.500000e+03	7.995000e+03	1.225250e+04	8.712000e+05

- Tiếp theo là kiểm tra xem có data nào có giá trị là Null ko : Data.isnull().sum()

- Loại bỏ một số cột không cần thiết như là cột date , id , zipcode

```
[42] # kiểm tra xem có giá trị null
     Data.isnull().sum()
     # bỏ đi 1 số cột ko cần thiết
     Data = Data.drop('date',axis=1)
     Data = Data.drop('id',axis=1)
     Data = Data.drop('zipcode',axis=1)
```

```
[43] X = Data.drop('price',axis =1).values
     y = Data['price'].values
     # tách ra 1 tập train và tấp test
     from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=101)
```

- Tiếp theo là tách thành 1 tập train và 1 tập test với thư viện train_test_split

- Ta sẽ dùng tập train để làm thuật toán cho dữ liệu học và kiểm tra model của chúng ta trên tập test

- Tiếp ta scale 2 tập X và y cho nhỏ lại

```
[44] # để scale số nhỏ lại
     y_train = y_train / 100000
     y_test = y_test / 100000
```

```
[45] #import standardscaler
     from sklearn.preprocessing import StandardScaler
     s_scaler = StandardScaler()
     X_train = s_scaler.fit_transform(X_train.astype(np.float))
     X_test = s_scaler.transform(X_test.astype(np.float))
```

- Tạo ra Neural Network Model

- Có 50 neuron , 4 lớp ẩn và 1 lớp output để dự đoán giá nhà

- Hàm kích hoạt là relu (lấy các giá trị lớn hơn 0)

- Thuật toán ADAM được dùng để tối ưu tính năng sai số (MAE)

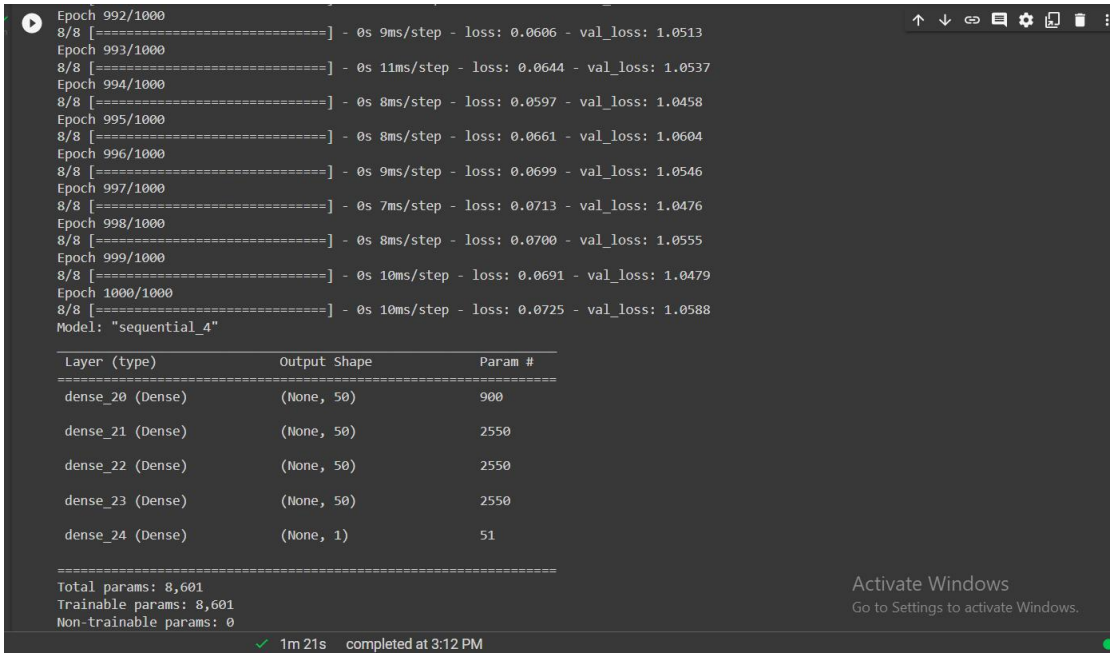
```
[55] # tạo neural network model
     from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import Dense, Activation
     from tensorflow.keras.optimizers import Adam

     # có 32 nhánh tương đương với số tính chất
     model = Sequential()
     model.add(Dense(50,activation='relu'))
     model.add(Dense(50,activation='relu'))
     model.add(Dense(50,activation='relu'))
     model.add(Dense(50,activation='relu'))
     model.add(Dense(1))
     model.compile(optimizer='Adam',loss='mae')
```

```
[56] model.fit(x=X_train,y=y_train,
                validation_data=(X_test,y_test),
                batch_size=128,epochs=2000)
     model.summary()
```

- Cho AI train với 1000 lần thử với mỗi lần thử là ghi sai số và sai số thực tế trong lịch sử train .

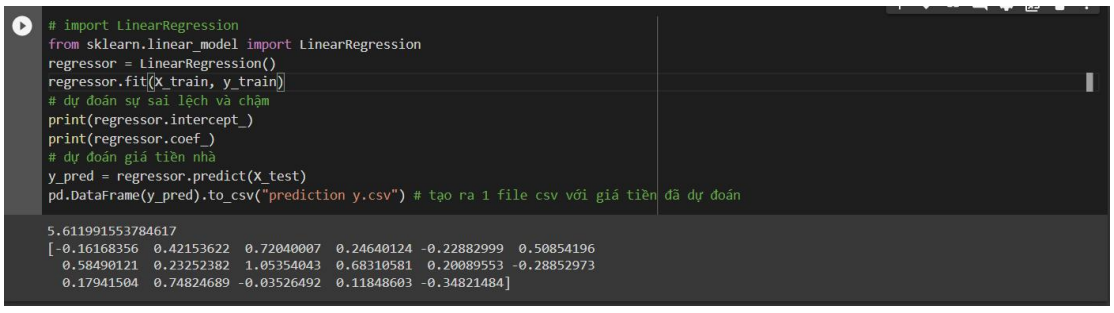
- Kết quả :



- Từ thư viện sklearn.linear_model import LinearRegression

- fit() thực hiện tính toán tối ưu hóa X_train , y_train

- In ra phần tử bị chặn lại với độ sai lệch



- So sánh giá tiền ban đầu và độ sai lệch của thuật toán (MAE - MSE - RMSE)
+) MAE : đo độ lớn trung bình của các lỗi

+) MSE : sai số bình phương trung bình giữa các giá trị được dự đoán và thực tế
+) RMSE : căn bậc 2 của mức trung bình của các sai số bình phương

```
[58] # so sánh số tiền test và số tiền đã dự đoán
     y_pred = regressor.predict(X_test)
     df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
     df1 = df.head(10)
     print(df1)
     # dự đoán sự sai sót trên 3 thuật toán MAE , MSE , RMSE
     from sklearn import metrics
     print('MAE:', metrics.mean_absolute_error(y_test, y_pred))
     print('MSE:', metrics.mean_squared_error(y_test, y_pred))
     print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
     print('VarScore:',metrics.explained_variance_score(y_test,y_pred))
```

```
    Actual  Predicted
0   4.6000   4.570868
1   2.2500   4.130399
2   2.2000   3.441875
3   4.5900   3.537252
4   6.1500  12.926026
5   4.5000   4.583219
6  13.7000  11.358632
7   5.5495   4.732338
8   2.4660   2.410174
9   5.2000   3.714555
MAE: 1.2736794294556337
MSE: 3.721773788819803
RMSE: 1.9291899307273515
VarScore: 0.7137144527579177
```

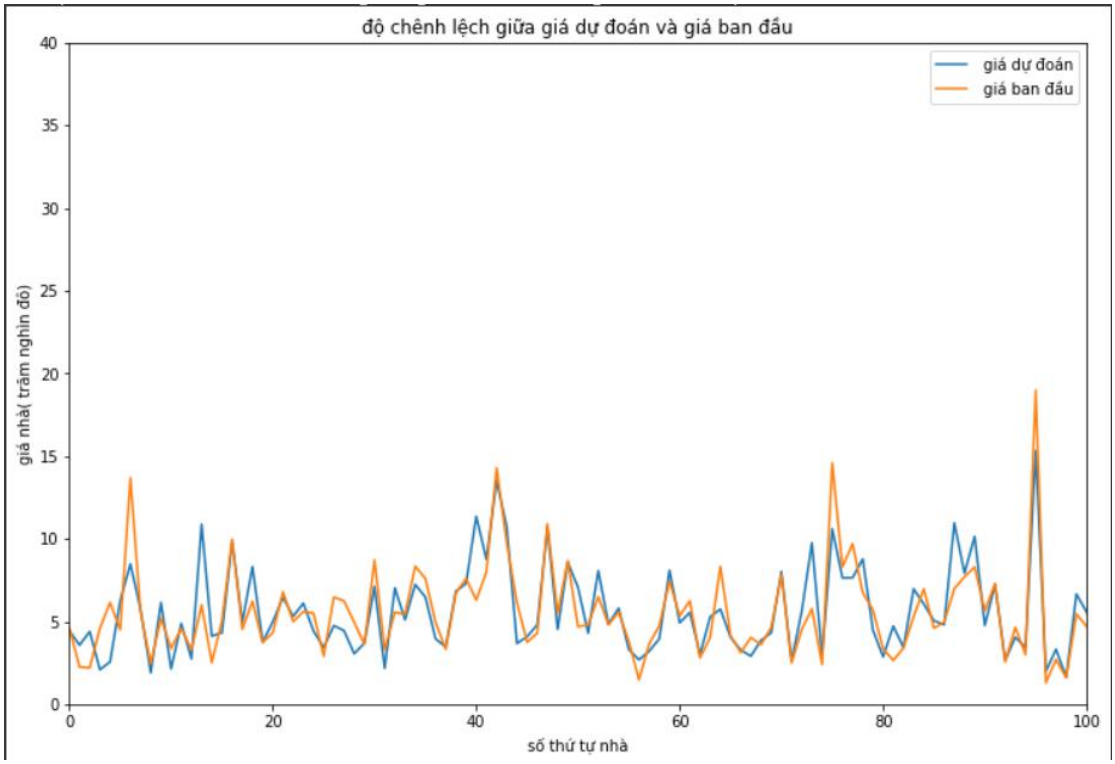


```
[59] y_pred = model.predict(X_test)
     from sklearn import metrics
     print('MAE:', metrics.mean_absolute_error(y_test, y_pred))
     print('MSE:', metrics.mean_squared_error(y_test, y_pred))
     print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
     print('VarScore:',metrics.explained_variance_score(y_test,y_pred))
```

```
16/16 [==============================] - 0s 2ms/step
MAE: 1.0715347824391837
MSE: 3.0598455345238365
RMSE: 1.7492414168787098
VarScore: 0.7645305232510322
```

Biểu đồ :

```
[60] fig, ax = plt.subplots(figsize=(12,8))
     ax.plot(y_pred,label=' giá dự đoán')
     ax.plot(y_test,label=' giá ban đầu')
     ax.legend(loc="upper right")
     ax.set_xlim([0,100])
     ax.set_ylim([0,40])
     plt.xlabel("số thứ tự nhà")
     plt.ylabel("giá nhà( trăm nghìn đô)")
     plt.title("độ chênh lệch giữa giá dự đoán và giá ban đầu")
```



- Dạng biểu đồ khác để thấy rõ hơn giữa y_predict và y test
+) màu xanh là y dự đoán
+) màu cam là y test

**) Kết luận
- Bằng cách vẽ biểu đồ ra ta thấy được giá dự đoán không xa với giá thực tế có thể giảm thiểu độ sai sót xuống bằng cách cho nhiều lớp hơn