Model_Train_Test

August 4, 2023

1 House Price Prediction

- Trong phần này, bạn sẽ học được cách sử dụng dữ liệu sẵn có để xây dựng một mô hình dự đoán (dự đoán giá nhà).
  - Xây dựng/ Khởi tạo mô hình như thế nào?
  - Đưa dữ liệu vào huấn luyện mô hình?
  - Sử dụng mô hình đã huấn luyện để dự đoán?
- Dữ liệu sử dụng: clean_data.csv (có được sau bài thực hành trên lớp)

2 Các bước tiến hành

2.1 Load dữ liệu và xử lí một vài bước trước khi huấn luyện

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

1. Import thư viện cần thiết

[2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[3]:
```
pd.set_option("display.max_columns", None)
pd.set_option("display.max_rows", None)
```

2. Load dữ liệu cần thiết

[ ]:
```
%cd /content/drive/MyDrive/ML_course/Preprocessing_practice/Practice/
↪Bangalore_House_Price_data
#Nếu chạy trên colab thì cũng cần trỏ tới thư mục phù hợp để lấy data
```

[ ]:
```
path = "./clean_data.csv"
df = pd.read_csv(path)
df.shape
```

[ ]:
```
df.head()
```

3. Phân chia dữ liệu train - test

[ ]:
```
# Xác định thông tin thuộc tính X và nhãn y
X = df.drop("price", axis=1)
y = df['price']
print('Shape of X = ', X.shape)
print('Shape of y = ', y.shape)
```

[ ]:
```
#Chia dữ liệu - train_test_split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,␣
 ↪random_state = 51)
print('Shape of X_train = ', X_train.shape)
print('Shape of y_train = ', y_train.shape)
print('Shape of X_test = ', X_test.shape)
print('Shape of y_test = ', y_test.shape)
```

4. Feature Scaling

   If feature scaling is not done, then a machine learning algorithm tends to weigh greater values, higher and consider smaller values as the lower values, regardless of the unit of the values.

[9]:
```
#Chuẩn hóa giá trị của các feature trong 1 phạm vi nào đó.

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
sc.fit(X_train)
X_train= sc.transform(X_train)
X_test = sc.transform(X_test)
```

2.2 Xây dựng mô hình huấn luyện

2.2.1 Linear Regression

[10]:
```
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error
lr = LinearRegression()
lr_lasso = Lasso()
lr_ridge = Ridge()
```

[11]:
```
def rmse(y_test, y_pred):
  return np.sqrt(mean_squared_error(y_test, y_pred))
```

[ ]:
```
lr.fit(X_train, y_train)
lr_score = lr.score(X_test, y_test) # with all num var 0.7842744111909903
```

```
lr_rmse = rmse(y_test, lr.predict(X_test))
lr_score, lr_rmse
```

[ ]:
```
# Lasso
lr_lasso.fit(X_train, y_train)
lr_lasso_score=lr_lasso.score(X_test, y_test) # with balcony 0.5162364637824872
lr_lasso_rmse = rmse(y_test, lr_lasso.predict(X_test))
lr_lasso_score, lr_lasso_rmse
```

[ ]:
```
# Ridge
lr_ridge.fit(X_train, y_train)
lr_ridge_score = lr_ridge.score(X_test, y_test) # with all num var 0.
↪7842744111909903
lr_ridge_rmse = rmse(y_test, lr_ridge.predict(X_test))
lr_ridge_score, lr_ridge_rmse
```

2.2.2 Support Vector Machine

[ ]:
```
from sklearn.svm import SVR
svr = SVR()
svr.fit(X_train,y_train)
svr_score=svr.score(X_test,y_test)
svr_rmse = rmse(y_test, svr.predict(X_test))
svr_score, svr_rmse
```

2.2.3 Random Forest

[ ]:
```
from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor()
rfr.fit(X_train,y_train)
rfr_score=rfr.score(X_test,y_test)
rfr_rmse = rmse(y_test, rfr.predict(X_test))
rfr_score, rfr_rmse
```

2.3 Test Model

[17]:
```
def␣
↪predict_house_price(model,bath,balcony,total_sqft_int,bhk,price_per_sqft,area_type,availabi
↪


 x =np.zeros(len(X.columns))

 x[0]=bath
 x[1]=balcony
 x[2]=total_sqft_int
 x[3]=bhk
```

```
  x[4]=price_per_sqft

  if "availability"=="Ready To Move":
    x[8]=1

  if 'area_type'+area_type in X.columns:
    area_type_index = np.where(X.columns=="area_type"+area_type)[0][0]
    x[area_type_index] =1

  if 'location_'+location in X.columns:
    loc_index = np.where(X.columns=="location_"+location)[0][0]
    x[loc_index] =1

  x = sc.transform([x])[0]

  return model.predict([x])[0]
```

area_type

availability

location

bath

balcony

total_sqft_int

bhk

price_per_sqft

Plot Area

Ready to Move

Devarabeesana

3

2

1672

3

8971.291866

[ ]:
```
# Test Linear Regression
lr_test = predict_house_price(model=lr,␣
 ↪bath=3,balcony=2,total_sqft_int=1672,bhk=3,price_per_sqft=8971.
 ↪291866,area_type="Plot  Area",availability="Ready To␣
 ↪Move",location="Devarabeesana Halli")
print("Test Linear Regression: ", lr_test)
```

```
# Test Lasso
lr_lasso_test = predict_house_price(model=lr_lasso,␣
 ↪bath=3,balcony=2,total_sqft_int=1672,bhk=3,price_per_sqft=8971.
 ↪291866,area_type="Plot  Area",availability="Ready To␣
 ↪Move",location="Devarabeesana Halli")
print("Test Lasso: ", lr_lasso_test)
```

[ ]:
```
# Test SVM
svm_test = predict_house_price(model=svr,␣
 ↪bath=3,balcony=2,total_sqft_int=1750,bhk=3,price_per_sqft=8571.
 ↪428571,area_type="Super built-up",availability="Ready To␣
 ↪Move",location="Devarabeesana Halli")
print("Test SVM: ", svm_test)
```

[ ]:
```
# Test Random Forest
test_random_forest =␣
 ↪predict_house_price(model=rfr,bath=3,balcony=3,total_sqft_int=1750,bhk=3,price_per_sqft=851
 ↪285714,area_type="Built-up Area",availability="Ready To␣
 ↪Move",location="Devarabeesana Halli")
print("Test Random Forest: ", test_random_forest)
```

2.4 Save model & load model

[ ]:
```
import joblib
joblib.dump(rfr, 'bangalore_house_price_prediction_model.pkl')
```

[22]:
```
bangalore_house_price_prediction_model = joblib.
 ↪load("bangalore_house_price_prediction_model.pkl")
```

[ ]:
```
predict_house_price(bangalore_house_price_prediction_model,bath=3,balcony=3,total_sqft_int=150
 ↪285714,area_type="Built-up Area",availability="Ready To␣
 ↪Move",location="Devarabeesana Halli")
```