REPORT

1) Đặt vấn đề: Cào dữ liệu từ trang web giá nhà, xử lý dữ liệu và xây dựng mô hình để dự đoán giá nhà.
2) Phương pháp:

- Data collecting: sử dụng beautifulsoup để thu thập dữ liệu từ trang web.
- Preprocessing: dữ liệu thu về có 1 số vấn đề:
    + Lượng dữ liệu bị khuyết khá nhiều -> giải pháp: xóa đi các hàng có data null.
    + Đơn vị ở cột Price lúc thu về là không đồng nhất bao gồm 5 loại (Tỷ, triệu,nghìn,thỏa thuận,NaN) -> đưa về cùng một đơn vị.
- Feature Engineer: từ title lọc ra được các giá trị Area, Floor, Bedroom, Bathroom và số lượng convenient gần đó. Đối với Area, Bedroom, Bathroom lọc ra từ content vì đôi lúc người ta chỉ đề cập trong title.
- EDA: làm rõ mối tương quan giữa các cột đối với cột Price.
- HeatMap:
    - **Định Vị Địa Lý với Geopy:**
        - Hàm fetch_coordinates nhận địa chỉ và một chuỗi đại diện người dùng làm đầu vào. Nó sử dụng công cụ Nominatim từ thư viện geopy để chuyển đổi địa chỉ thành tọa độ địa lý là kinh độ và vĩ độ (latitude and longitude)
        - Hàm xử lý các ngoại lệ như GeocoderTimedOut và GeopyError để đảm bảo tính robust trước các vấn đề định vị địa lý phổ biến như hết thời gian chờ và lỗi.
        - Nó bao gồm một độ trễ (time.sleep(1)) giữa các yêu cầu để tuân thủ giới hạn tốc độ của dịch vụ Nominatim, ngăn người dùng bị chặn do yêu cầu quá mức.
    - **Tạo Bản Đồ Nhiệt:**
        - Dữ liệu đã lọc được chuyển thành danh sách các cặp [vĩ độ, kinh độ], sau đó được sử dụng để tạo một lớp bản đồ nhiệt.
        - Một đối tượng bản đồ folium (map_hcm) được tạo với vị trí trung tâm được đặt tại một điểm trong TP.HCM và một mức độ phóng cụ thể.
        - Lớp bản đồ nhiệt được thêm vào đối tượng bản đồ này, trực quan hóa mật độ của các vị trí được cung cấp trong TP.HCM.
    - **Hiển Thị Bản Đồ:**
        - Cuối cùng, kịch bản hiển thị bản đồ với lớp phủ bản đồ nhiệt. Trực quan này có thể được sử dụng để phân tích các mô hình địa lý, mật độ, hoặc phân phối của các điểm dữ liệu trong khu vực đã chỉ định của TP.HCM.
- Build Model: sử dụng 4 model là: Random Forest, …….

+ **Random Forest**: mô hình này thu được tỷ lệ chính xác dao động khoảng 50%.
    - Mô hình Random Forest được lựa chọn cho dự án này vì khả năng xử lý tốt các tập dữ liệu lớn và phức tạp, cũng như khả năng giảm thiểu overfitting.
    - Các tham số chính của mô hình bao gồm:
        - ➢ n_estimators: Số lượng cây trong rừng. Thường thì số lượng cây càng lớn thì mô hình càng chính xác, nhưng cũng cần cân nhắc thời gian huấn luyện.
        - ➢ max_depth: Độ sâu tối đa của mỗi cây. Điều này giúp kiểm soát độ phức tạp của mô hình và tránh overfitting.
        - ➢ min_samples_split và min_samples_leaf: Số lượng mẫu tối thiểu cần thiết để chia một nút và số lượng mẫu tối thiểu ở một lá (leaf) của cây.
        - ➢ max_features: Số lượng đặc trưng được xem xét khi tìm kiếm chia tốt nhất tại mỗi nút.
    - Mô hình được đánh giá trên tập dữ liệu kiểm tra bằng cách sử dụng các chỉ số đánh giá như Mean Absolute Error (MAE), Mean Squared Error (MSE), và Root Mean Square Error(RMSE).
    - Tối ưu hóa hiệu suất của mô hình: điều chỉnh tham số (hyperparameter tuning) bằng cách sử dụng Random Search trên các tham số chính như n_estimators, max_depth, và max_features. Kết quả sau khi điều chỉnh tham số cho thấy mô hình có thể đạt được hiệu suất cao hơn, với các chỉ số MAE, MSE và RMSE cải thiện đáng kể. Điều này chứng tỏ tầm quan trọng của việc điều chỉnh tham số trong việc tối ưu hóa mô hình Random Forest.
+ **Ridge Regression**, còn được biết đến với tên gọi Tikhonov regularization, là một phương pháp hồi quy được sử dụng trong thống kê để phân tích dữ liệu đa biến. Phương pháp này giúp giải quyết vấn đề đa cộng tuyến trong hồi quy tuyến tính bằng cách thêm một thuật ngữ điều chỉnh (regularization term) vào hàm mất mát. Mục tiêu là tối thiểu hóa tổng của bình phương sai số dự đoán và bình phương tổng của các hệ số, nhằm giảm thiểu sự phụ thuộc vào dữ liệu và tăng cường độ ổn định của mô hình.

Huấn Luyện Mô Hình**:** Sử dụng tập dữ liệu huấn luyện để điều chỉnh các tham số của mô hình, bao gồm cả tham số điều chỉnh alpha, nhằm tối thiểu hóa hàm mất mát. Ban đầu, huấn luyện mô hình với tham số alpha=50, kết quả cho thấy rằng điểm giữa tập X_test và y_test là 0.3814 ,giữa tập X_train và y_train là 0.357 là mức điểm khá thấp và R2 Score: 0.381

- Vậy nên áp dụng công cụ GridSeachCV để tìm ra tham số alpha tốt nhất. Xác định một lưới các giá trị khả thi cho tham số alpha phạm vi rất nhỏ từ (1e-15) đến rất lớn (20). Việc sử dụng công cụ GridSearchCV với mô hình Ridge, lưới siêu tham số đã xác định, sử dụng điểm số là 'neg_mean_squared_error' cho việc đánh giá, và số lượng fold trong cross-validation là 5.
- Việc sử dụng cross-validation với cv=5 giúp đảm bảo rằng kết quả tinh chỉnh siêu tham số không bị ảnh hưởng bởi sự phân chia ngẫu nhiên cụ thể nào của dữ liệu.
- Sau khi dùng công cụ GridSearchCV, tìm được tham số alpha tốt nhất là 0.01. Đối với tham số alpha này, điểm số giữa tập test và train tăng lên lần lượt là 0.414 và 0.448 và R2 score = 0.448

+ **Gradient Boosting**:

Gradient Boosting là một phương pháp học máy mạnh mẽ được sử dụng rộng rãi cho các bài toán phân loại và hồi quy. Phương pháp này xây dựng một mô hình dự đoán mạnh bằng cách kết hợp nhiều mô hình yếu, thường là cây quyết định. Các bước làm:

- Khởi tạo giá trị ban đầu:

```
model_gbr = ensemble.GradientBoostingRegressor(n_estimators = 400,
max_depth = 5, min_samples_split = 2,
        learning_rate = 0.1, loss = 'squared_error')
```

- Ta nhận được mô hình bị overfitting

Ta điều chỉnh lại các siêu tham số. Ta dùng RandomizedSearchCV để điều chỉnh lại các siêu tham số. Ta được kết quả cuối cùng có giá trị tốt nhất như sau:

```
model_gbr_best_esti =
ensemble.GradientBoostingRegressor(learning_rate=0.05,
max_depth=7, max_features=2, min_samples_leaf=9,
min_samples_split=6, n_estimators=250, subsample=0.85)
```

- Ta nhận được giá trị tốt hơn là

```
Tap train
R2:  0.8016259007473153
MSE:  1.2938250691089 34e+18
MAE:  809592677.5875525


----------------------------------------
```

```
Tap test
R2:  0.5278209268240812
MSE:  3.3212306147123343e+18
MAE:  1259847591.0714514
```

Nhận xét về vô hình:

Lợi ích:

- Hiệu quả cao: Gradient Boosting thường đạt hiệu quả dự đoán cao trên nhiều loại dữ liệu khác nhau.
- Linh hoạt: Có thể sử dụng cho cả bài toán phân loại và hồi quy.
- Khả năng điều chỉnh: Có nhiều tham số để điều chỉnh như số lượng cây, độ sâu của cây, và hệ số học để tối ưu hóa mô hình.

Nhược điểm:

- Dễ bị overfitting: Nếu không kiểm soát tốt các tham số, Gradient Boosting có thể dễ dàng bị overfitting.
- Tốn thời gian: Quá trình huấn luyện có thể tốn nhiều thời gian do tính toán lặp đi lặp lại.
- Phức tạp trong việc điều chỉnh tham số: Đòi hỏi nhiều công sức trong việc tìm kiếm và tối ưu hóa các tham số mô hình.

+ **Decision Tree là một trong những thuật toán học máy phổ biến và dễ hiểu nhất.**

Quy trình huấn luyện :

- Sử dụng mô hình decision tree với các tham số mặc định trong sklearn , và em nhận thấy mô hình bị overfit do tập r2 score nhỏ hơn rất nhiều so với tập test (0.07<0.99)
- Sau đó dùng Pre-pruning để cắt tỉa cây nhằm giảm độ phức tạp của cây và trách overfit . Em đã dùng công cụ là GridSearchCV để tìm ra bộ tham số tốt nhất cho (max_depth, min_samples_split, min_samples_leaf) với tiêu chí đánh giá là neg_mean_sqared_error và số lượng fold trong cross-validation là 5 thì em nhận được mô hình tốt nhất có r2 Score là 0.4 trên tập test và 0.51 trên tập train . Có thể nói nó tuy vẫn còn overfit

nhưng đã đỡ hơn rất nhiều so với lần đầu tiên và nó cũng có độ chính xác cao hơn nhiều so với mô hình đầu .

- Cuối cùng dùng Post-pruning để cắt tỉa cây và thử nghiệm xem giá alpha tốt nhất là bao nhiêu và vẽ đồ thị xem xét khả năng của model thì em thấy giá trị r2_score của tập test cao nhất có thể là khoảng 0.4 nên có thể đây là con số tốt nhất .

3) Đánh giá: Mặc dù đã chọn 4 model để dự đoán nhưng hiệu quả thu được dao động không quá 50% với model thu được kết quả tốt nhất là Gradient Boosting khoảng 52.7% trên tập test.
4) Kết luận:

Mặc dù đã áp dụng nhiều phương pháp khác nhau, bao gồm Random Forest, Ridge Regression, Gradient Boosting, và Decision Tree, hiệu quả dự đoán giá nhà vẫn chưa đạt được mức độ chính xác cao, với mô hình Gradient Boosting đạt R2 score cao nhất khoảng 52.7% trên tập test. Điều này cho thấy dữ liệu hiện tại và các đặc trưng sử dụng chưa đủ mạnh để mô hình có thể dự đoán giá nhà một cách chính xác. Để cải thiện kết quả, cần bổ sung thêm dữ liệu và khám phá các kỹ thuật mới trong xử lý và tối ưu hóa mô hình.

Việc điều chỉnh và tối ưu hóa các tham số của mô hình đã cho thấy tầm quan trọng của việc này trong việc cải thiện hiệu suất. Tuy nhiên, các mô hình vẫn có xu hướng overfitting, do đó, cần tiếp tục nghiên cứu và áp dụng các phương pháp ngăn chặn overfitting hiệu quả hơn.