

DỰ ĐOÁN DOANH SỐ BÁN HÀNG THÔNG QUA CÁC MÔ HÌNH HỌC MÁY

Thuật toán Linear Regression, K-Nearest Neighbors, Decision Tree, Random Forest



NỘI DUNG CHÍNH

- 01** Giới thiệu tổng quan đề tài
- 02** Cơ sở lý thuyết và các nghiên cứu liên quan
- 03** Mô hình dự đoán doanh thu
- 04** Kết quả thực nghiệm và đề xuất
- 05** Kết luận và hướng phát triển



1. GIỚI THIỆU TỔNG QUAN ĐỀ TÀI

Lý do chọn đề tài

- Xây dựng chiến lược phát triển
- Giảm chi phí, cải thiện doanh số
- Dự báo cảm tính dựa trên kinh nghiệm
- Phụ thuộc nhân sự dự đoán

Mục tiêu đề tài

- Cách thức chuyển đổi dữ liệu
- Phân tích khai phá dữ liệu, tìm insight
- Xác định yếu tố ảnh hưởng đến Sales
- Xác định mô hình dự đoán hiệu quả nhất

Phương pháp nghiên cứu

- Phương pháp xử lý dữ liệu
- Phương pháp mô tả, khai thác dữ liệu
- Phương pháp học máy



2. CƠ SỞ LÝ THUYẾT VÀ CÁC NGHIÊN CỨU LIÊN QUAN

2.1. Giới thiệu Python

- Python là ngôn ngữ lập trình bậc cao để lập trình đa dạng, lập trình hướng đối tượng có cấu trúc dữ liệu cấp cao với hệ thống thư viện lớn.
- Giúp phân tích dữ liệu dễ dàng, thuận tiện hơn

Phân tích dữ liệu với R/Python

- Bước 1: Thiết lập môi trường Python
- Bước 2: Tìm hiểu các khái niệm cơ bản về Python
- Bước 3: Hiểu hoạt động Python
- Bước 4: Thực hành làm việc với tập dữ liệu



2.2. Dự đoán doanh số

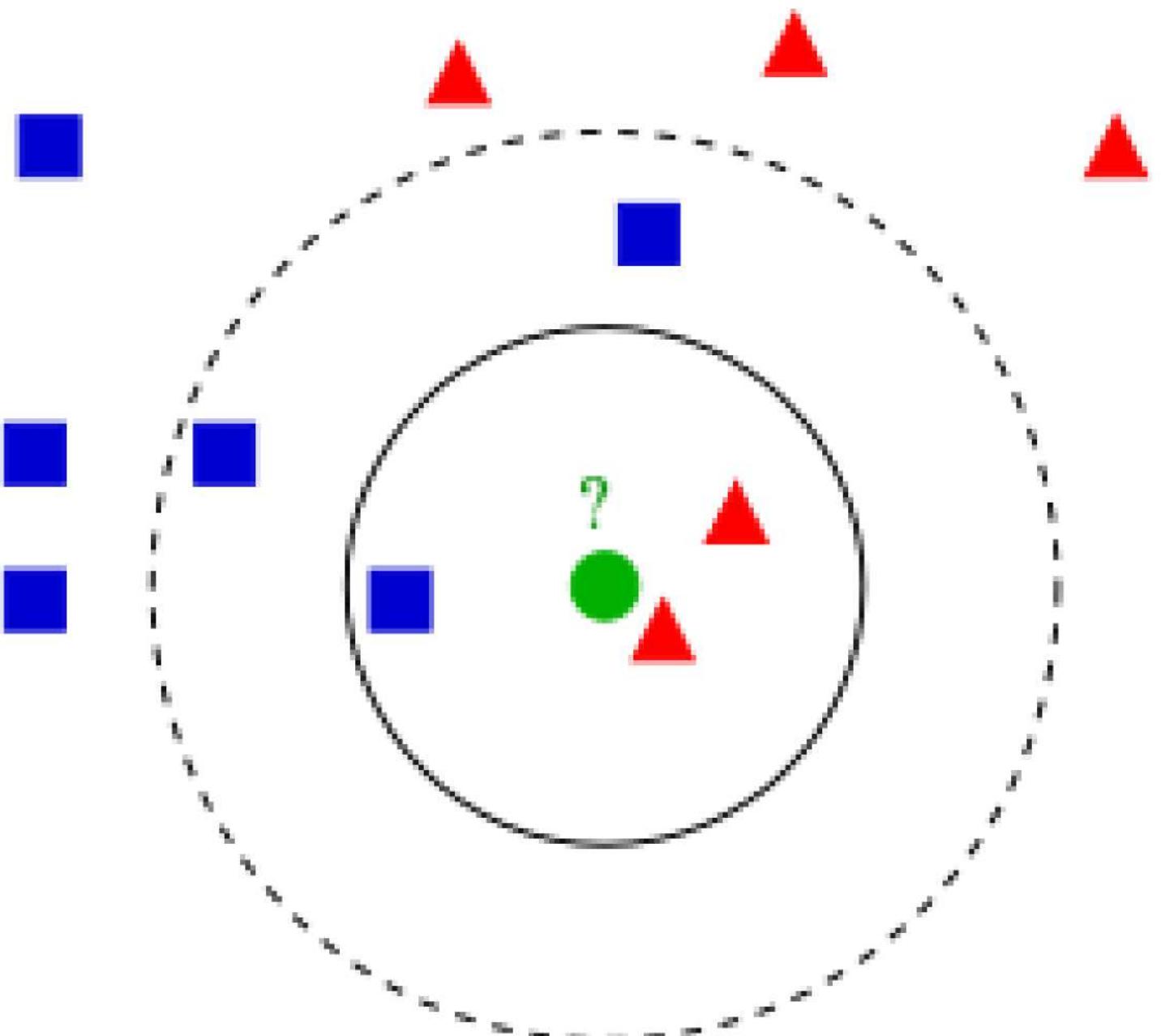
- Ước tính về chỉ số bán hàng trong tương lai. Điều kiện cần để dự đoán doanh số chuẩn: Báo cáo bán hàng, báo cáo thị phần độ phủ, báo cáo thị trường
- Tầm quan trọng: Thiết lập duy trì sản xuất, cơ sở đưa ra quyết định, cam kết hoạt động bán hàng
- Yếu tố ảnh hưởng: Đối thủ, thay đổi công nghệ, hành động Chính phủ, yếu tố sản xuất
- Thuật toán: Linear Regression, Decision Tree, Logistic Regression, K-nearest Neighbor, Random Forest, Stepwise Regression,...



2.3. Lý thuyết ứng dụng trong đề tài

- K-Nearest Neighbors

- Khái niệm: Tìm ra đầu ra của dữ liệu dựa trên thông tin của những dữ liệu training gần nó nhất.
- Ý tưởng: Tìm đầu ra của một điểm giá trị mới bằng cách tính toán khoảng cách với k điểm dữ liệu gần nó nhất.



- Khoảng cách không gian trong vector

Distance functions

Euclidean

$$\sqrt{\sum_{i=1}^k (x_i - y_i)^2}$$

Manhattan

$$\sum_{i=1}^k |x_i - y_i|$$

Minkowski

$$\left(\sum_{i=1}^k (|x_i - y_i|)^q \right)^{1/q}$$

- Chỉ số, siêu tham số quan trọng

- n_neighbors: int, default=5
- metric: str or callable, default='minkowski'
- p: int, default=2
- weights: {'uniform', 'distance'}, callable or None, default='uniform'
- n_jobs: int, default=None
- algorithm: {'auto', 'ball_tree', 'kd_tree', 'brute'}, default='auto'
- leaf_size: int, default=30

Nhược điểm

Giá trị k cần được tính toán kỹ càng

Nhiều dung lượng, thời gian thực hiện

Ảnh hưởng lời nguyền chiều dữ liệu

Phương pháp cải thiện

Mẹo chọn k:

- Chọn $k = \sqrt{2}n$
- Thực hiện thay đổi k và đánh giá từng mô hình

Phương pháp giảm chiều dữ liệu bao gồm cả feature selection và dimensionality reduction

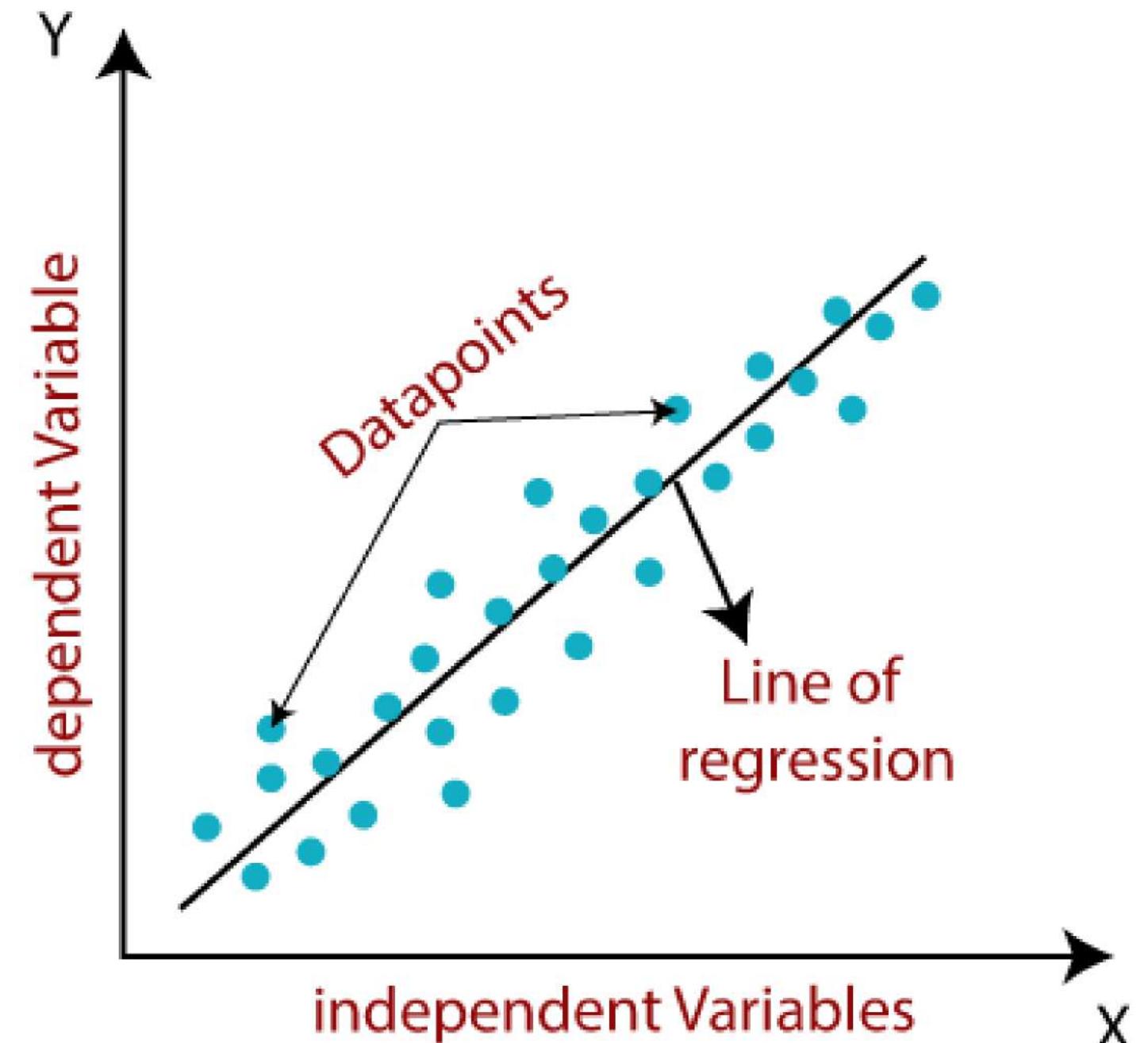
Khó áp dụng lên tập dữ liệu phân loại dữ liệu không phải số, tính toán phức tạp

2.3. Lý thuyết ứng dụng trong đề tài

- Linear Regression

- Khái niệm: Thuật toán hồi quy tuyến tính thuộc nhóm học có giám sát. Hồi quy dữ liệu giữa biến phụ thuộc có giá trị liên tục với một hoặc nhiều biến độc lập có giá trị liên tục hoặc phân loại.

- Ý tưởng: Tìm một đường thẳng để đường thẳng đó gần với tất cả các điểm trên đồ thị của chúng ta nhất có thể.



Simple Linear Regression

- Mối quan hệ giữa 1 biến phụ thuộc và 1 biến độc lập
- Phương trình: $Y = \beta_0 + \beta_1 X$

Trong đó:

- β_0, β_1 là các hằng số
- Y: Biến phụ thuộc (biến mục tiêu)
- X: Biến độc lập
- β_1 : Độ dốc của đường hồi quy (weight)
- β_0 : Hệ số chận (intercept) (bias)

Multiple Linear Regression

- Mối quan hệ giữa 1 biến phụ thuộc và nhiều biến độc lập
- Phương trình:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n + \varepsilon$$

Trong đó:

- ε : Sai số (phần dư)

Hàm chi phí (Cost Function)

- *Mean Squared Error (MSE)*

- Sai số bình phương trung bình giữa các giá trị được dự đoán và giá trị thực tế
- Tính chất: Không âm, càng thấp thì mô hình có tính chính xác càng cao

- *Root Mean Squared Error (RMSE)*

- Căn bậc hai mức trung bình của các sai số bình phương
- Tính chất: Không âm, càng thấp thì độ tin cậy của mô hình càng cao

- *Mean Absolute Error (MAE)*

- Đo độ lớn trung bình của các lỗi. Là trung bình tổng trị tuyệt đối sai số giữa đầu ra dự đoán và kết quả thực

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

$$MAE = \frac{\sum_{i=1}^n |y_i - x_i|}{n}$$

Thuật toán Gradient Descent

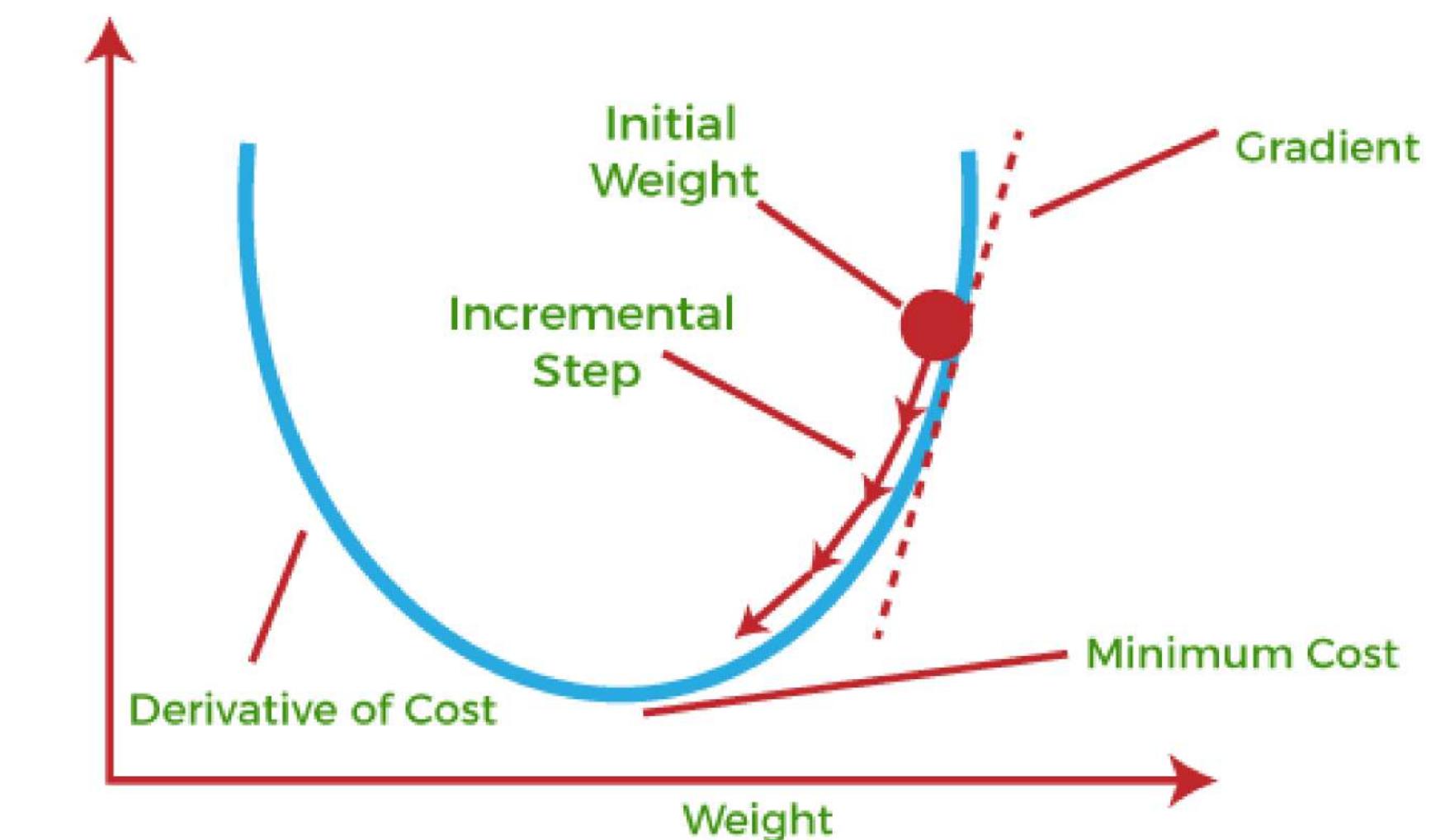
- Tối ưu hóa hàm chi phí, tìm ra các trọng số tối ưu
- Thuật toán tìm giá trị nhỏ nhất của hàm số $f(x)$ dựa trên đạo hàm

+ Bước 1: Khởi tạo giá trị $x=x_0$ tùy ý

+ Bước 2: Gán $x = x - \text{learning_rate} * f'(x)$

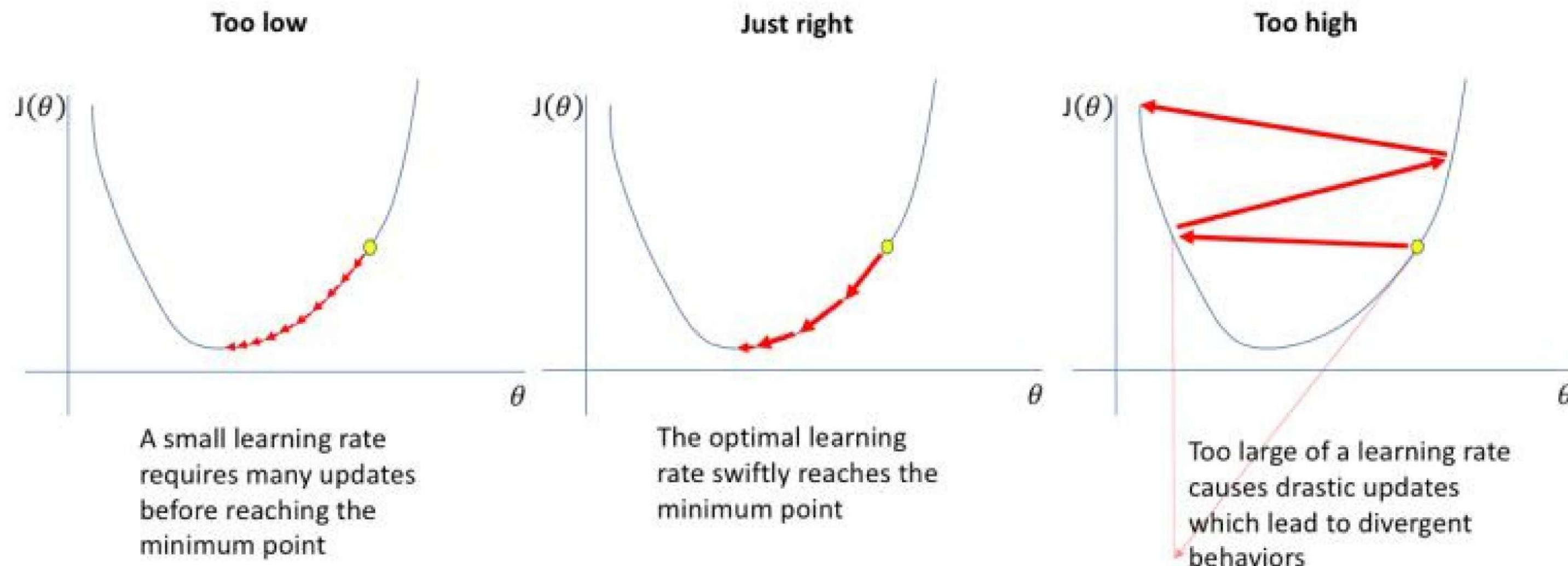
+ Bước 3: Tính lại $f(x)$. Nếu $f(x)$ đủ nhỏ thì dừng

lại, ngược lại tiếp tục bước 2



Thuật toán Gradient Descent

- 3 trường hợp của learning rate



Thuật toán Gradient Descent

- Áp dụng thuật toán Gradient Descent vào hàm chi phí, các giá trị được cập nhật trong mỗi lần lặp lại. Chúng ta sẽ thực hiện thuật toán Gradient Descent cho đến khi tìm được các hệ số của phương trình tối ưu (weight và bias)

$$B_0 = B_0 - \alpha \cdot \frac{2}{n} \sum_{i=1}^n (pred_i - y_i)$$

$$B_1 = B_1 - \alpha \cdot \frac{2}{n} \sum_{i=1}^n (pred_i - y_i) \cdot x_i$$

Một vài chỉ số quan trọng của mô hình

- **fit_intercept: bool, default=True**

Việc tính toán hệ số chặn cho mô hình, nếu đặt bằng false thì sẽ coi hệ số chặn bằng 0. Tham số này được mặc định là True.

- **copy_X: bool, default=True**

Việc quyết định việc sao chép (True) hoặc ghi đè các biến đầu vào (False). Tham số này được mặc định là True.

- **n_jobs: int, default=None**

Đại diện cho số lượng công việc được sử dụng trong tính toán song song (nếu như có nhiều hơn 1 biến mục tiêu). None thường có nghĩa là một công việc và -1 sử dụng tất cả các bộ xử lý. Tham số này được mặc định là None.

- **positive: bool, default=False**

Nếu chuyển thành True, nó sẽ bắt buộc các hệ số phải dương. Mặc định của tham số này là False.

Nhược điểm

Phương pháp cải thiện

Nhạy cảm với các giá trị ngoại lai, dữ liệu nhiễu

Xử lý giá trị ngoại lai trước khi chạy

Vấn đề overfitting
(low bias, high variance)

Để giải quyết vấn đề này, chúng ta, có các cách sau:

- K-fold cross-validation
- Nếu dữ liệu huấn luyện nhỏ thì thêm dữ liệu sạch và phù hợp hơn
- Nếu dữ liệu huấn luyện quá lớn, hãy thực hiện một số lựa chọn tính năng và xóa các tính năng không cần thiết.
- Regularization

Dễ xảy ra hiện tượng đa cộng tuyến

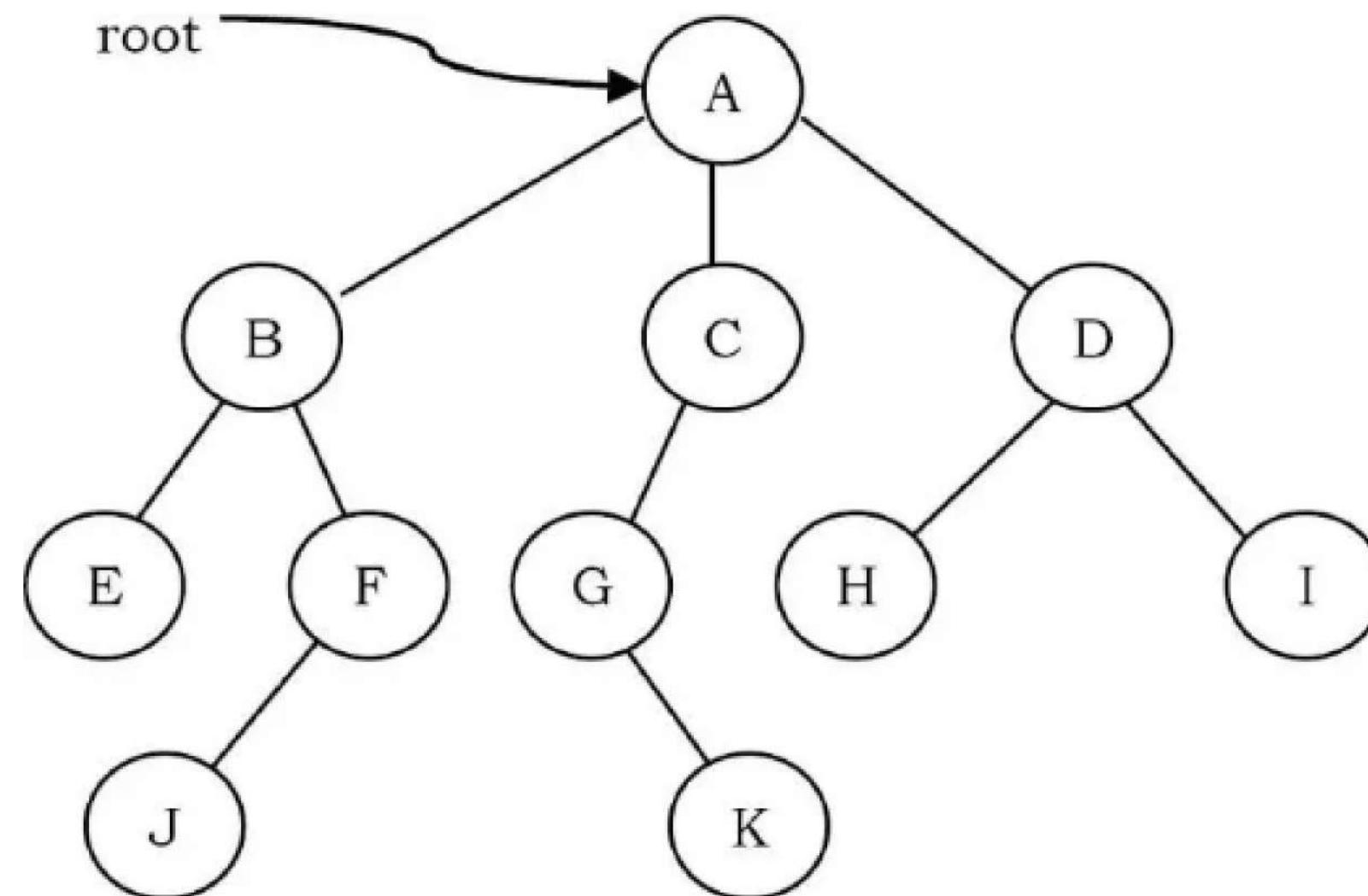
Dựa vào ma trận tương quan hoặc đo lường giá trị VIF. Sau đó:

- Loại bỏ biến độc lập có hệ số VIF vượt qua giá trị tiêu chuẩn
- Thu thập thêm mẫu dữ liệu nếu mẫu nhỏ
- Hủy bỏ dữ liệu, điều chỉnh mô hình (Nếu vấn đề xuất phát từ việc chọn mô hình)

2.3. Lý thuyết ứng dụng trong đề tài

- Decision Tree

- Khái niệm: Là một loại học máy có giám sát được sử dụng để phân loại hoặc đưa ra dự đoán dựa vào việc trả lời một loạt các câu hỏi khác nhau.



- Các chỉ số ASM phổ biến

Đối với bài toán hồi quy, hiệu quả của sự phân tách nhánh dựa vào mức độ chênh lệch giữa dự đoán và kết quả thực tế. Mức độ chênh lệch này thường được thể hiện qua chỉ số ASM là Mean Square Error (MSE). Công thức của MSE như sau:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

Trong đó:

- Y là giá trị biến phụ thuộc thực tế
- \hat{Y} là giá trị biến phụ thuộc được dự đoán từ mô hình
- n là số mẫu của bộ dữ liệu đầu vào

Bên cạnh MSE, cây quyết định hồi quy còn có thể sử dụng các chỉ số ASM khác như Friedman_MSE, MAE, Poisson variance.

- Các chỉ số, siêu tham số quan trọng

- criterion: {"squared_error", "friedman_mse", "absolute_error", "poisson"},
default="squared_error"
- splitter: {"best", "random"}, default="best"
- max_depth: int, default=None
- min_samples_split: int hoặc float, default=2
- min_samples_leaf: int hoặc float, default=1
- max_features: int, float or {"auto", "sqrt", "log2"}, default=None
- random_state: int, giá trị RandomState hoặc None, default=None
- max_leaf_nodes: int, default=None
- min_impurity_decrease: float, default=0.0

Phương pháp cải thiện

- Đặt ra một tiêu chuẩn dừng - Pruning

Pruning là quá trình cắt tỉa một cây quyết định để nó trở nên đơn giản và bao quát hơn. Trước khi bắt đầu pruning, một cây quyết định sẽ được xây dựng để phân lớp đúng mọi điểm trong tập huấn luyện. Điều này cũng đồng nghĩa với việc đặt ra một số tiêu chuẩn dừng cho cây quyết định.

- Ensemble methods: Random forest

Ensemble methods là phương pháp tổng hợp kết quả dự đoán của nhiều model để đưa ra model cuối cùng. Đối với cây quyết định, ensemble methods sẽ tập hợp kết quả của nhiều cây quyết định để đưa ra một kết quả tối ưu nhất.

Đối với các nhiệm vụ phân loại, đầu ra của Random forests là loại được chọn bởi hầu hết các cây. Đối với các tác vụ hồi quy, giá trị trung bình hoặc dự đoán trung bình của các cây riêng lẻ được trả về.

- Feature selection hoặc dimensionality reduction

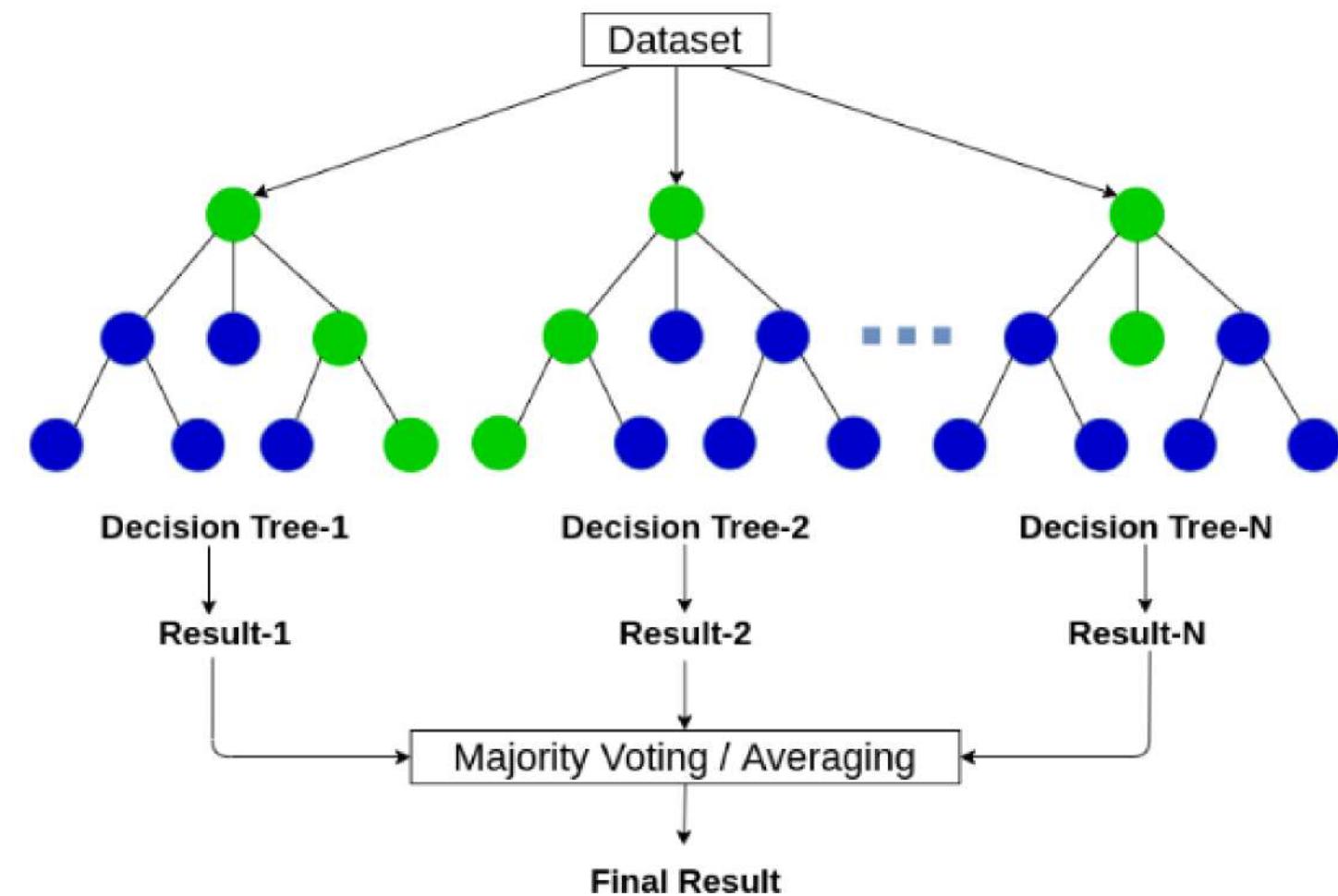
Feature selection và dimensionality reduction được sử dụng để giảm số lượng feature trong tập dữ liệu.

2.3. Lý thuyết ứng dụng trong đề tài

- Random Forest

- Khái niệm: Là một thuật toán học máy có giám sát dùng để phục vụ các mục đích phân loại, hồi quy bằng cách xây dựng và kết hợp đầu ra của nhiều cây quyết định để đạt được một kết quả duy nhất.

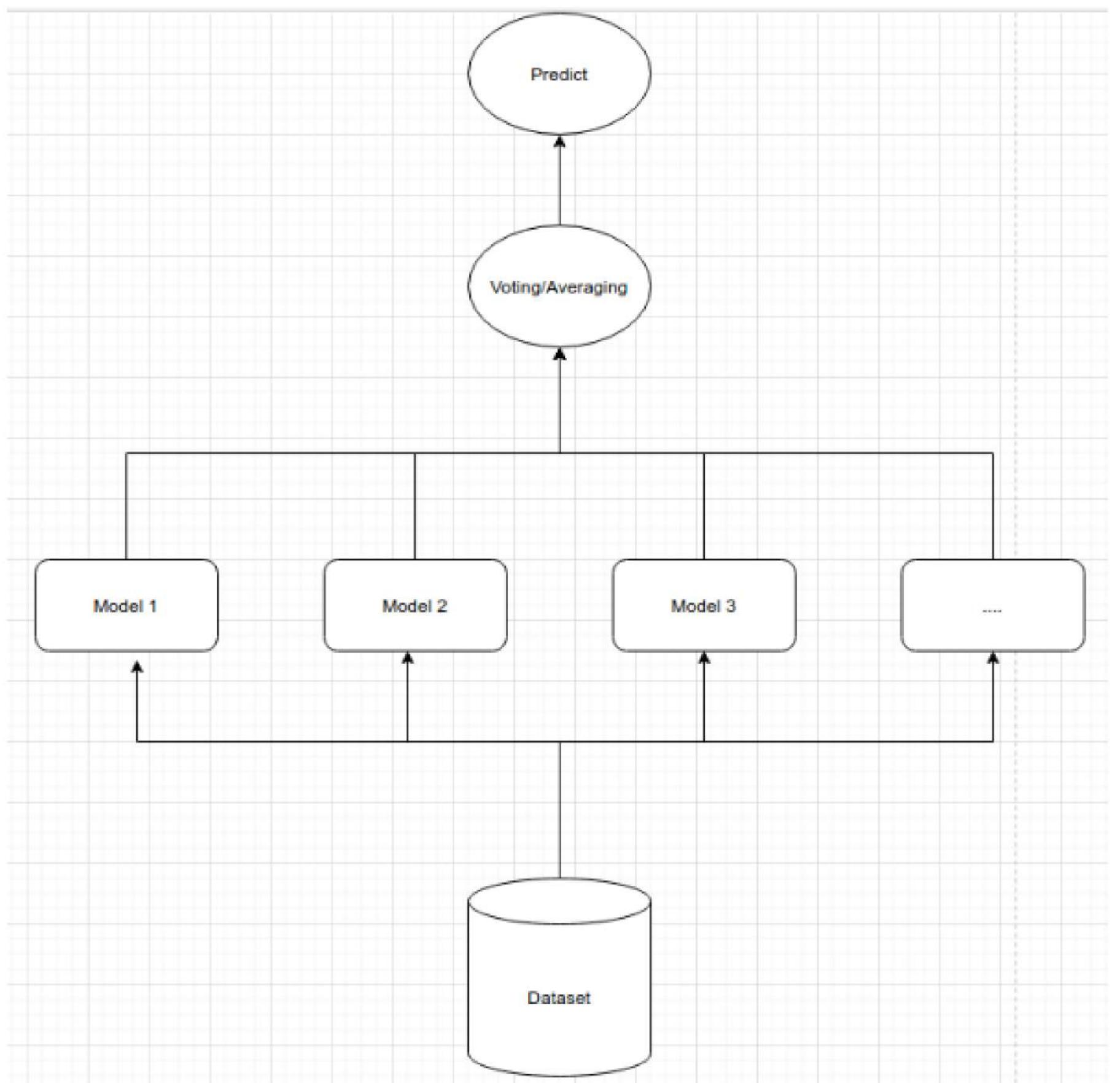
- Ý tưởng: Một tập hợp nhiều cây quyết định sẽ được tạo ra dựa trên các mẫu dữ liệu được chọn ngẫu nhiên, các kết quả từ các cây quyết định sẽ được tổng hợp lại lấy giá trị trung bình hoặc bỏ phiếu đa số để chọn ra cây quyết định cuối cùng đưa ra kết quả tốt nhất.



Mô hình Random Forest hoạt động dựa trên sự phối hợp giữa mô hình kết hợp (ensembling) và quá trình lấy mẫu tái lập (bootstrapping).

- Ensemble methods: Ensemble methods sẽ tập hợp kết quả của nhiều cây quyết định (Decision Tree) để đưa ra một kết quả tối ưu nhất.

Các thuật toán ensemble có thể được chia thành 03 nhóm chính là: bagging, boosting và stacking.



- Xây dựng thuật toán Random Forest:

- + Chọn T là số lượng các cây quyết định thành phần trong mô hình Random Forest sẽ được xây dựng.
- + Từ tập dữ liệu D , lấy mẫu tái lập Bootstrapping m dữ liệu ngẫu nhiên để tạo thành một tập dữ liệu con.
- + Tiếp tục, chọn ra ngẫu nhiên n thuộc tính để được bộ dữ liệu mới gồm m dữ liệu và mỗi dữ liệu có n thuộc tính.
- + Dùng thuật toán Decision Tree để xây dựng và huấn luyện cây quyết định với bộ dữ liệu được tạo thành ở bước 2 và 3. Lặp lại T lần để xây dựng và huấn luyện T cây quyết định cho mô hình Random Forest.
- + Thực hiện bầu cử đa số hoặc lấy giá trị trung bình giữa các cây quyết định để đưa ra kết quả cuối cùng.

- Các chỉ số, siêu tham số quan trọng

- n_estimators
- bootstrap=True
- max_depth
- criterion
- n_jobs
- max_features
- min_samples_leaf
- max_leaf_nodes
- max_samples
- class_weight
- min_impurity_split
- min_samples_split
- random_state
- oob_score

Nhược điểm

Phương pháp cải thiện

Chậm tạo dự đoán

Điều chỉnh tham số kết hợp sử dụng GridSearchCV hoặc RandomizeSearchCV để tìm ra mô hình nào tối ưu nhất.

Hạn chế với các lớp mất cân bằng

Để giải quyết vấn đề mất cân bằng dữ liệu, số nhãn lớp, có một số phương pháp như sau: Under sampling, over sampling, thay đổi metric đánh giá

Khó giải quyết được những vấn đề có dữ liệu phụ thuộc thời gian liên tục, tuyến tính

Sử dụng Feature selection (Filter, Wrapper, Embedded), Dimensionality reduction (Linear discriminant analysis - LDA và principal components analysis - PCA) để giảm số lượng feature trong tập dữ liệu.

Không giải tổng quát hóa các trường hợp với dữ liệu hoàn toàn mới trong bài toán hồi quy

2.4. Phương pháp đánh giá mô hình

- Mean Absolute Error - MAE

$$\text{MAE} = \frac{\sum_{i=1}^n |y_i - x_i|}{n}$$

- Mean Squared Error - MSE

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

- Root Mean Square Error - RMSE

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^N (x_i - \hat{x}_i)^2}{N}}$$

- Root Mean Square Percentage Error - RMPSE

Xác định lỗi phần trăm bình phương trung bình gốc

$$RMPSE = \sqrt{\frac{1}{n} \sum_{i=1}^n \left(\frac{y_i - \hat{y}_i}{y_i} \right)^2}$$

2.4. Phương pháp đánh giá mô hình

- R Squared - R²

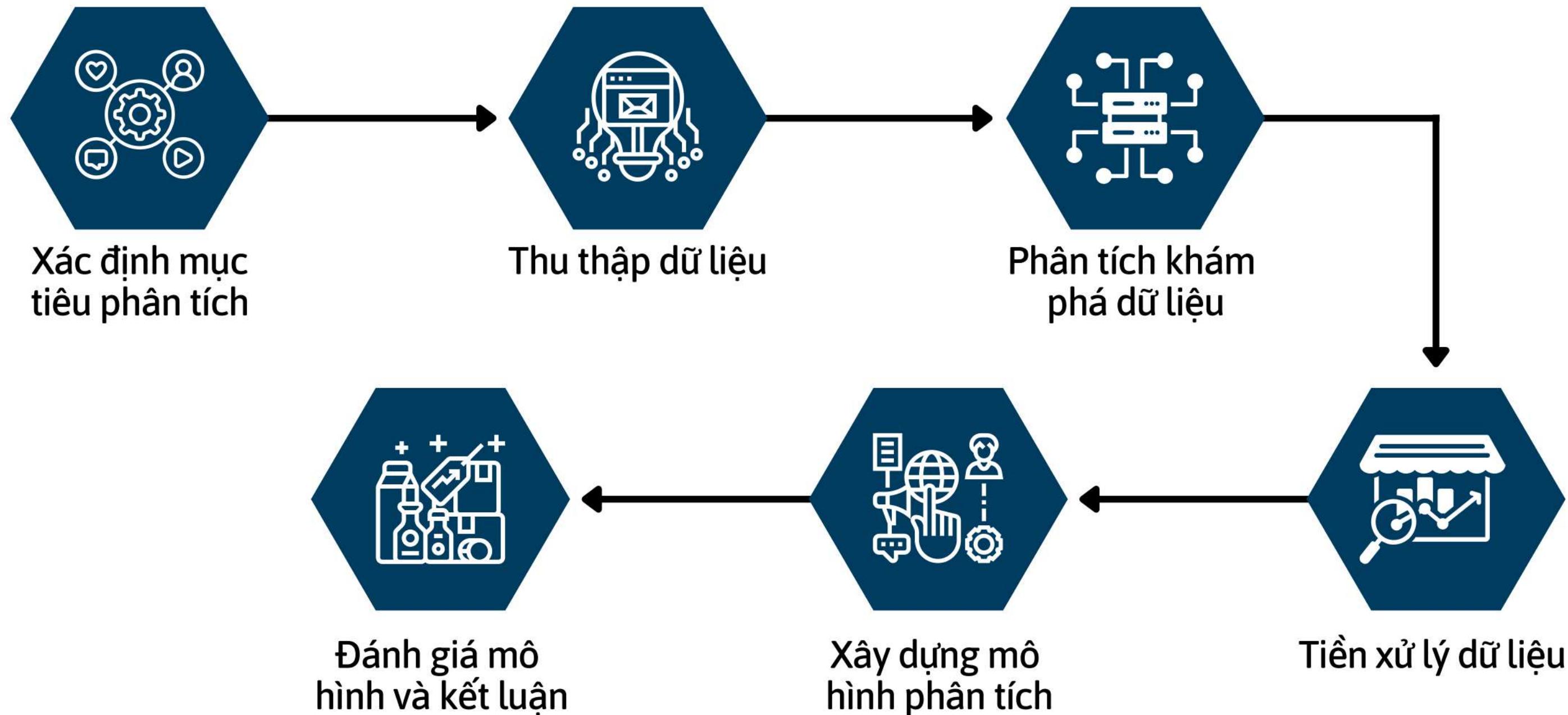
R squared phản ánh mức độ phù hợp của mô hình, được tính bằng công thức:

$$R^2 = 1 - \frac{ESS}{TSS}$$

Chỉ số R squared càng cao thì mô hình càng chính xác.

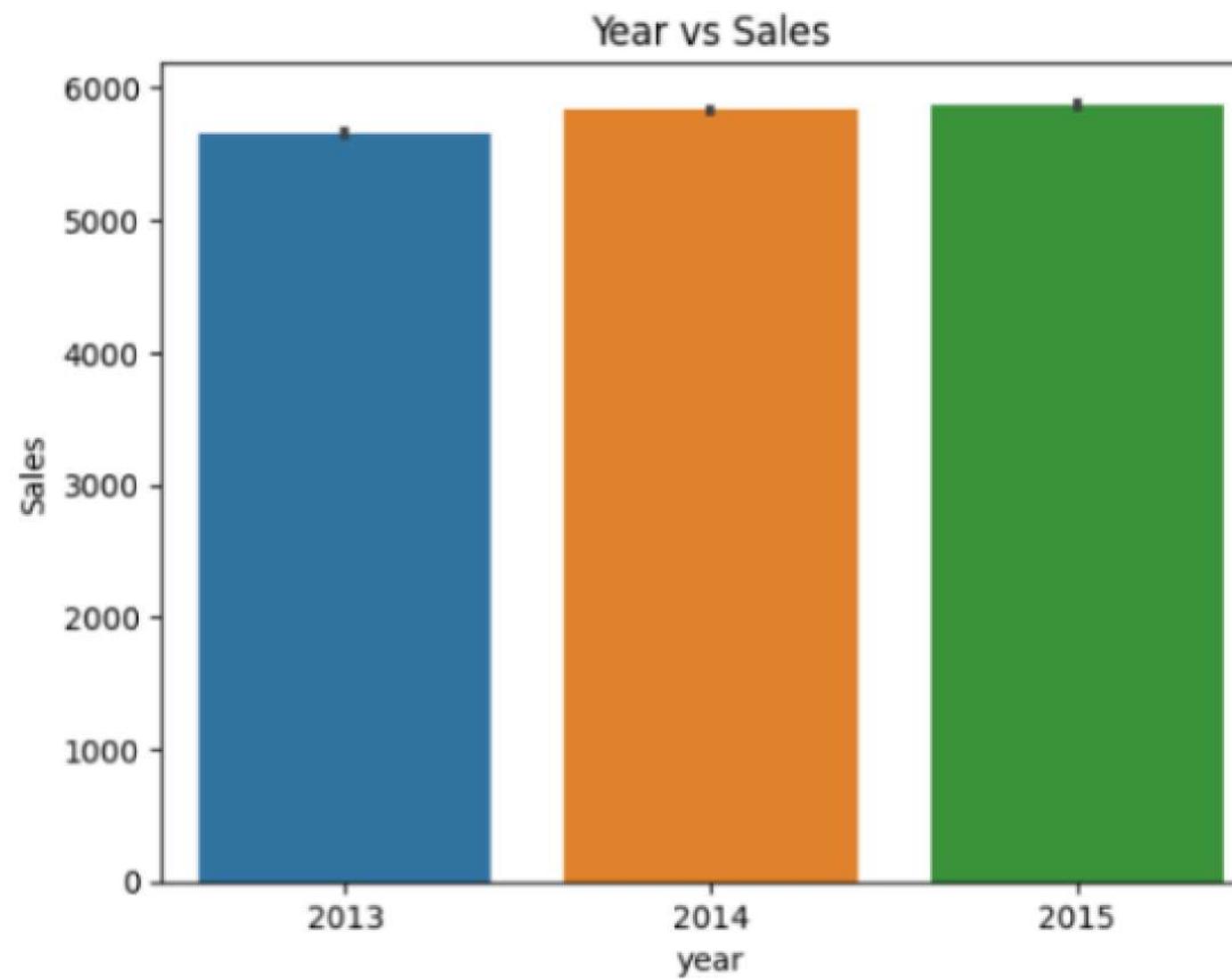
3. MÔ HÌNH DỰ ĐOÁN DOANH THU

3.1. Quy trình phân tích và xây dựng mô hình

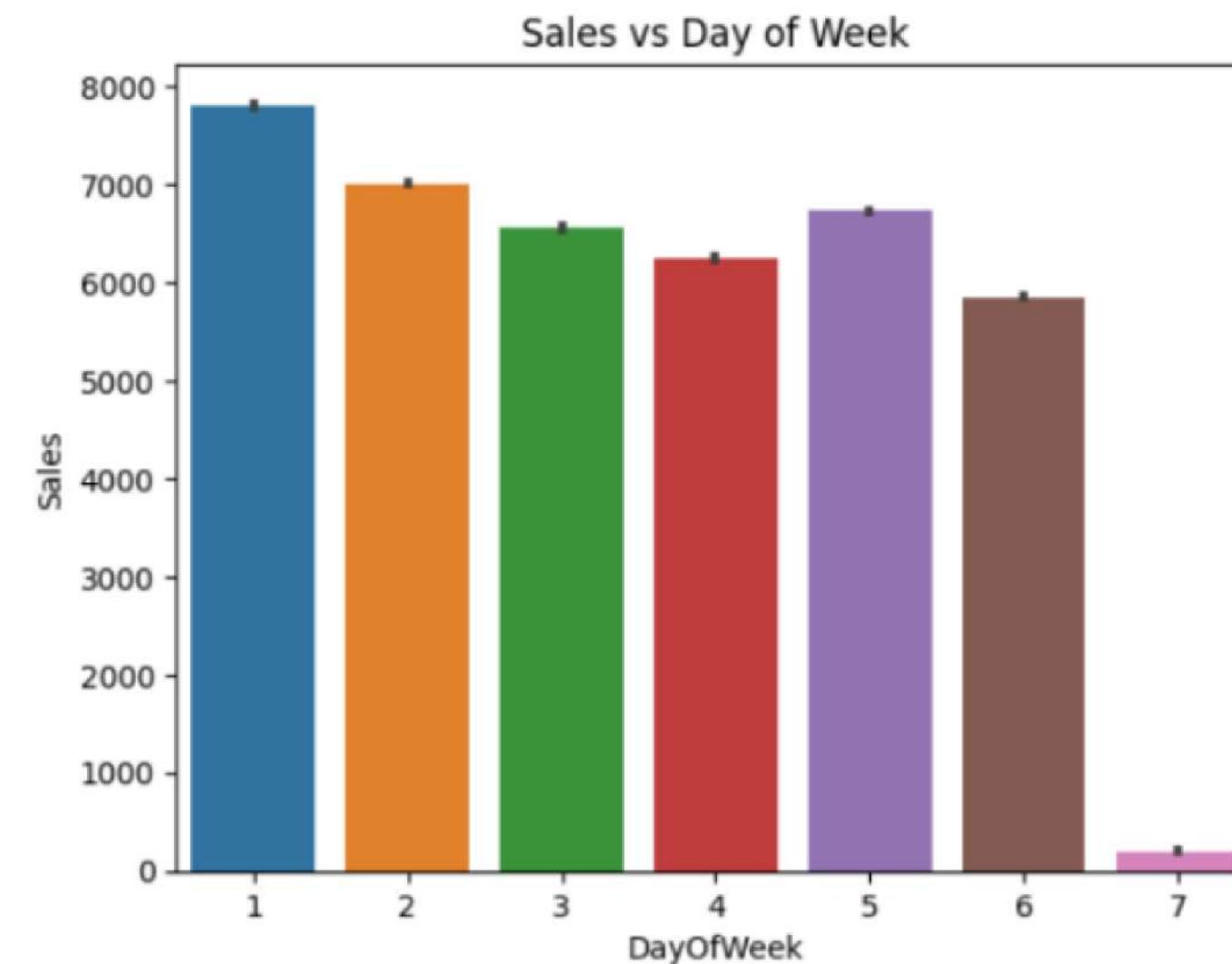


3.2. Phân tích khám phá dữ liệu - EDA

- Trực quan hóa dữ liệu



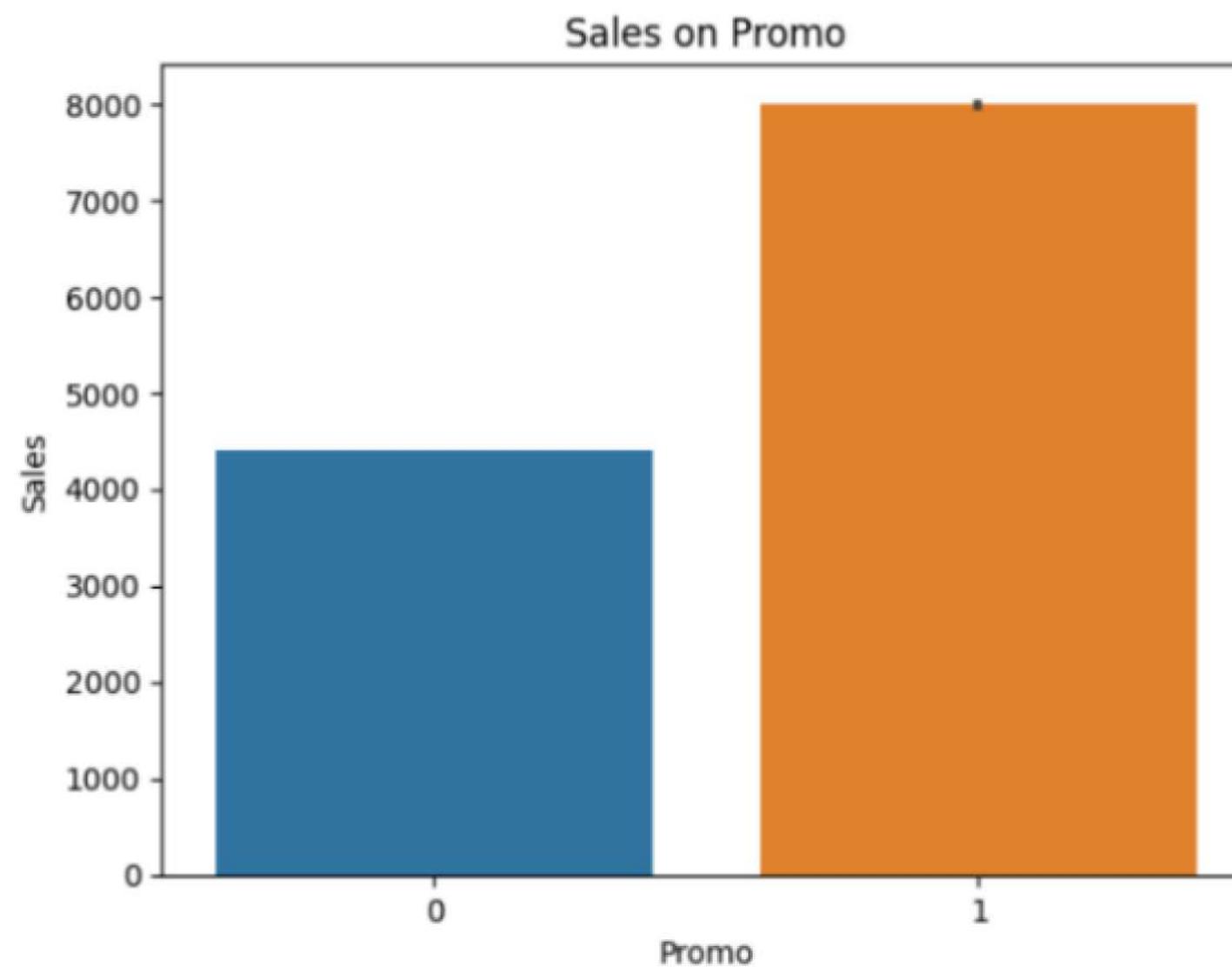
Biểu đồ thể hiện doanh thu theo năm



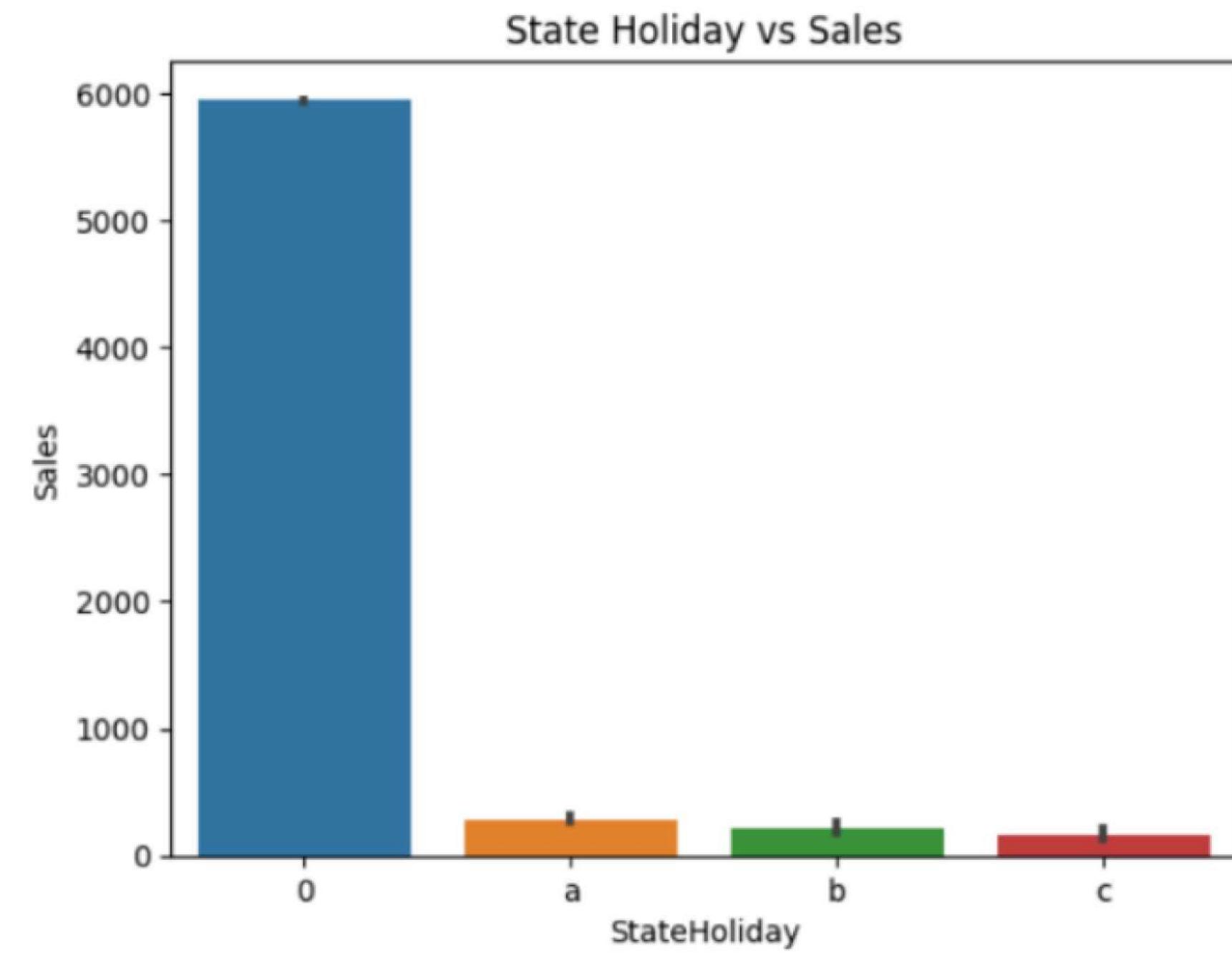
Biểu đồ thể hiện doanh thu theo ngày trong tuần

3.2. Phân tích khám phá dữ liệu - EDA

- Trực quan hóa dữ liệu



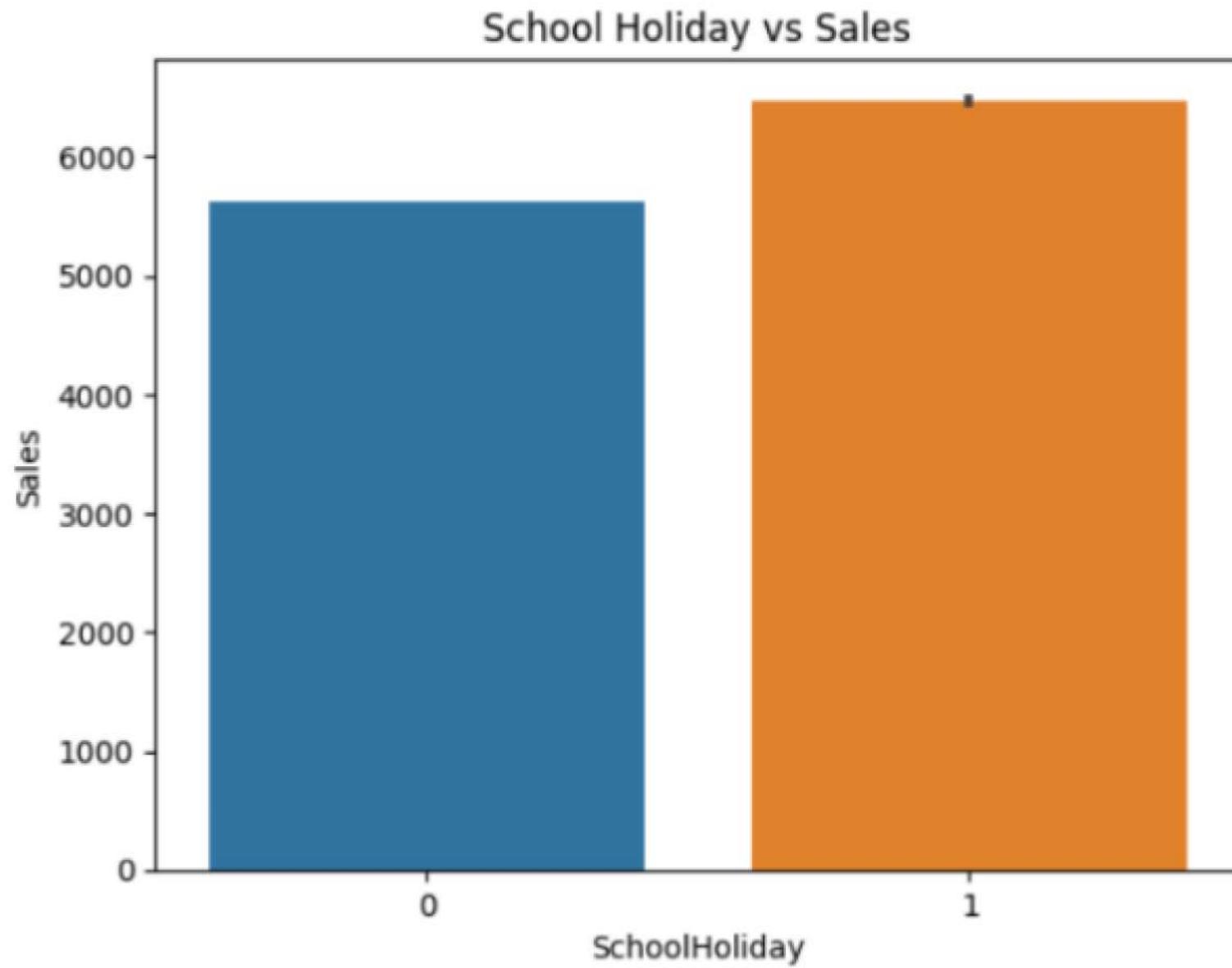
Biểu đồ thể hiện doanh thu theo khuyến mãi



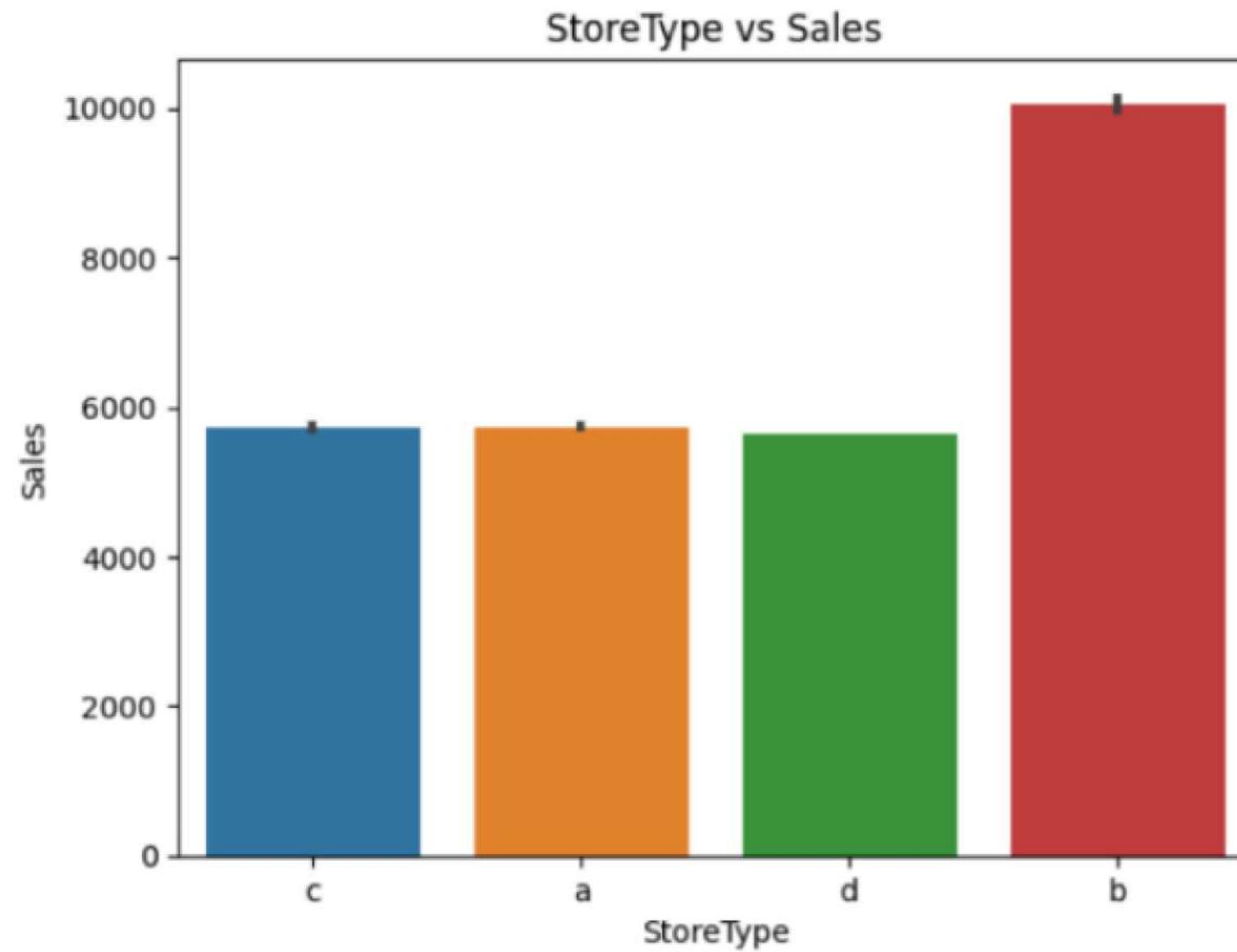
Biểu đồ thể hiện doanh thu theo ngày lễ

3.2. Phân tích khám phá dữ liệu - EDA

- Trực quan hóa dữ liệu



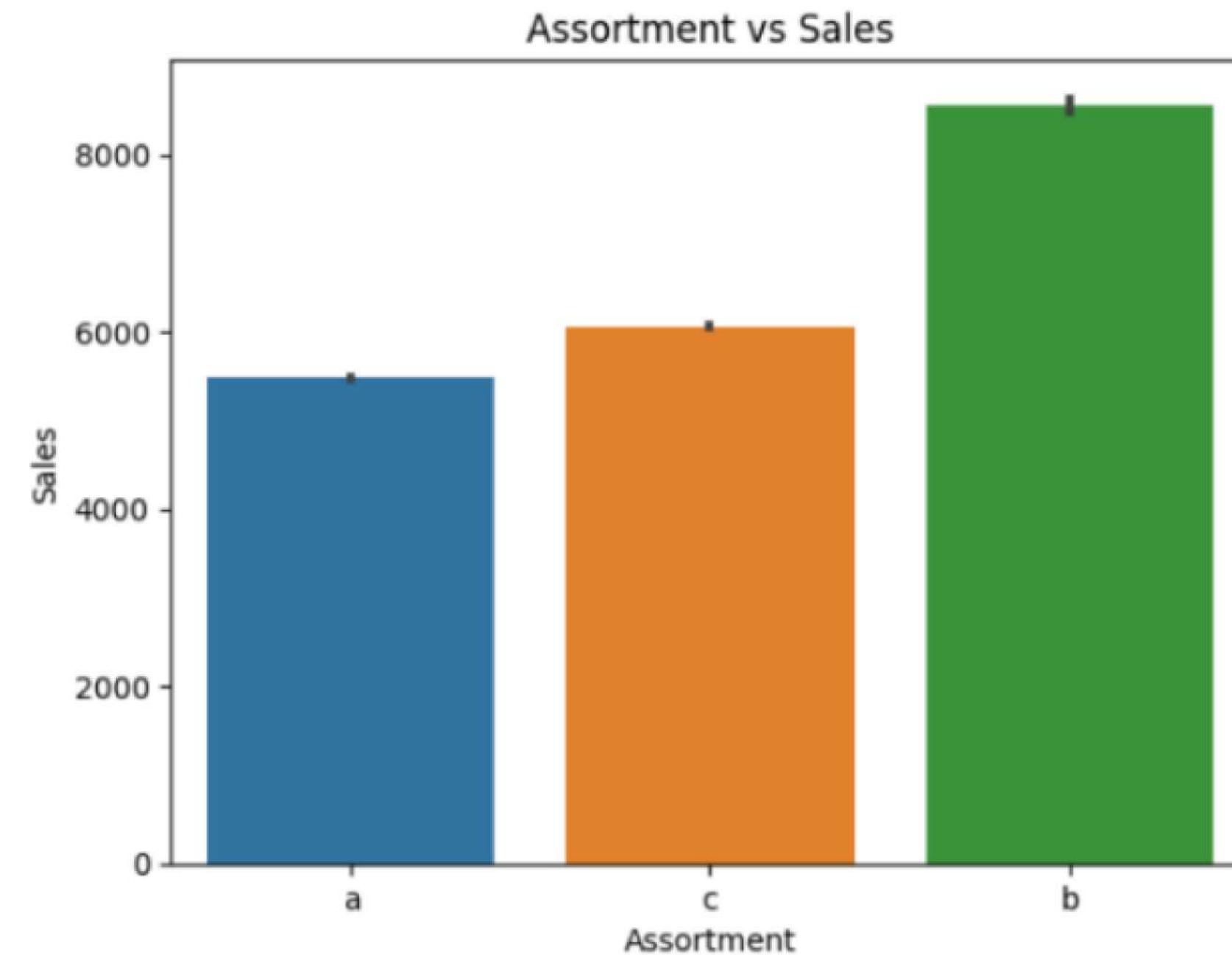
Biểu đồ thể hiện doanh thu theo ngày nghỉ ở trường



Biểu đồ thể hiện doanh thu theo loại cửa hàng

3.2. Phân tích khám phá dữ liệu - EDA

- Trực quan hóa dữ liệu



Biểu đồ thể hiện doanh thu theo Assortment

3.2. Phân tích khám phá dữ liệu - EDA

- Xử lý giá trị null

Kiểm tra giá trị null ở file store và file train.

```
store_data.isnull().sum()
```

Store	0
StoreType	0
Assortment	0
CompetitionDistance	3
CompetitionOpenSinceMonth	354
CompetitionOpenSinceYear	354
Promo2	0
Promo2SinceWeek	544
Promo2SinceYear	544
PromoInterval	544
dtype: int64	

```
train_data.isnull().sum()
```

Store	0
DayOfWeek	0
Date	0
Sales	0
Customers	0
Open	0
Promo	0
StateHoliday	0
SchoolHoliday	0
dtype: int64	

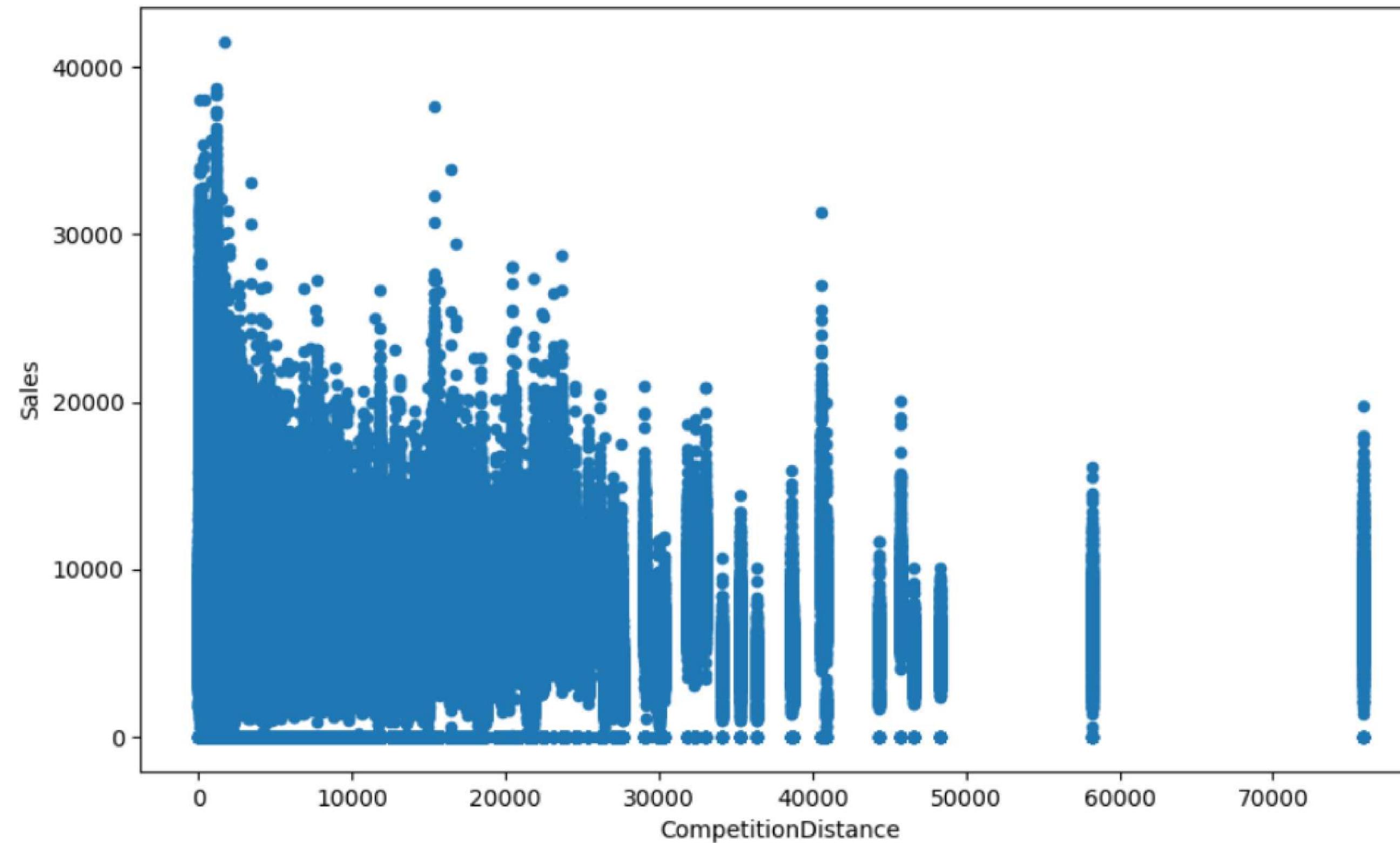
3.2. Phân tích khám phá dữ liệu - EDA

- Xử lý giá trị null

Tiến hành fill null

- Giá trị null ở những cột Promo2SinceWeek, Promo2SinceYear, Promointerval được fill null bằng 0.
- Giá trị null ở cột CompetitionDistance được fill null bằng giá trị trung bình.
- Giá trị null ở cột CompetitionOpenSinceMonth, CompetitionOpenSinceYear được điền bằng giá trị tháng, năm hiện tại

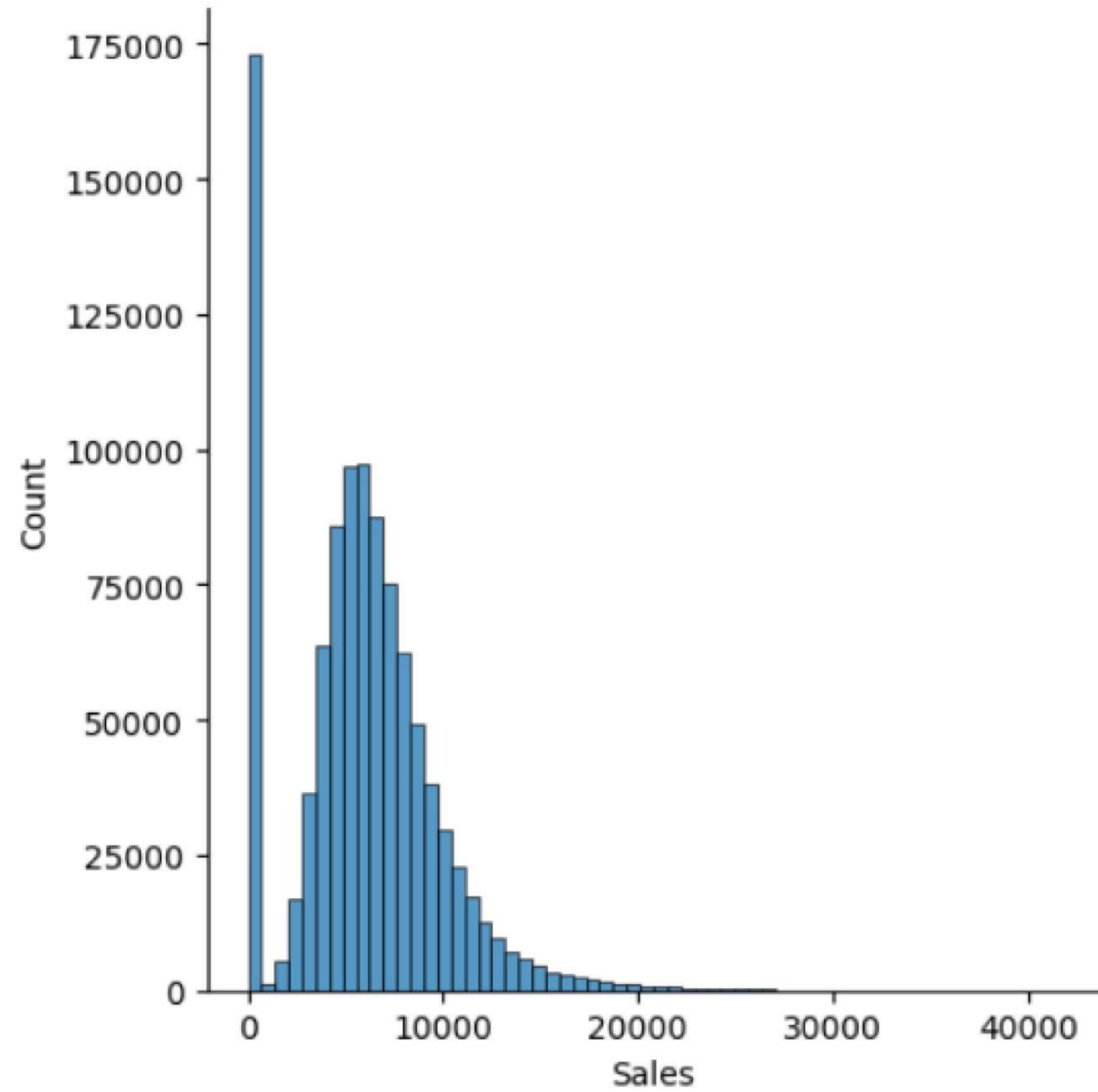
3.2. Phân tích khám phá dữ liệu - EDA



Biểu đồ thể hiện doanh thu theo CompetitionDistance

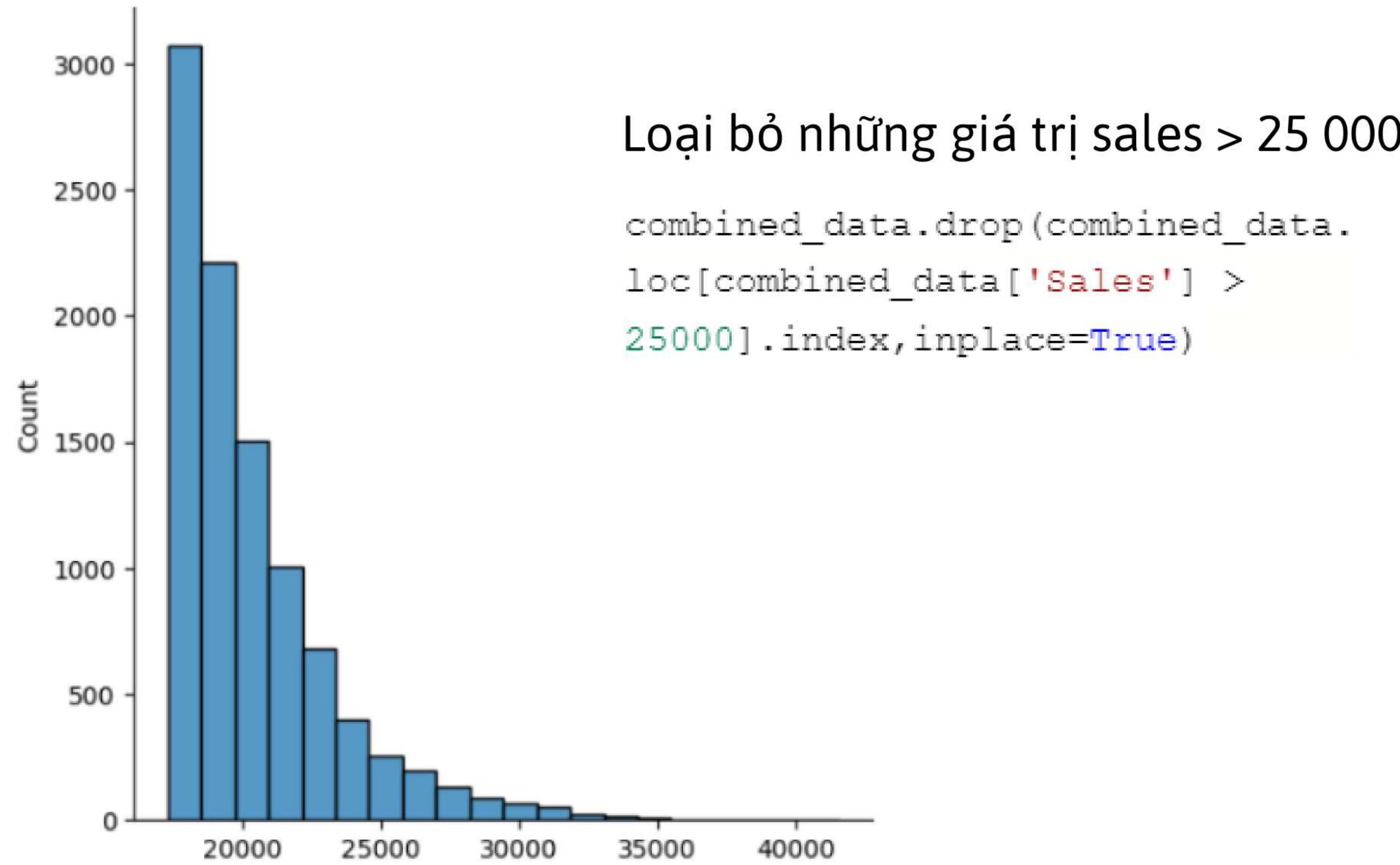
3.2. Phân tích khám phá dữ liệu - EDA

- Xử lý giá trị ngoại lai



3.2. Phân tích khám phá dữ liệu - EDA

- Xử lý giá trị ngoại lai



Loại bỏ những giá trị sales > 25 000

```
combined_data.drop(combined_data.  
loc[combined_data['Sales'] >  
25000].index,inplace=True)
```

Biểu đồ phân phối của những giá trị ngoại lai

3.3. Tiền xử lý dữ liệu

- Tiến hành tách cột ‘Date’ thành 2 cột ‘Year’ và ‘Month’, đồng thời xóa cột ‘Date’.

Store	int64	Sales	int64
StoreType	object	Customers	int64
Assortment	object	Open	int64
CompetitionDistance	float64	Promo	int64
CompetitionOpenSinceMonth	float64	StateHoliday	object
CompetitionOpenSinceYear	float64	SchoolHoliday	int64
Promo2	int64	Year	int64
Promo2SinceWeek	float64	Month	int64
Promo2SinceYear	float64	dtype: object	
PromoInterval	object		
DayOfWeek	int64		

3.3. Tiền xử lý dữ liệu

- Chuyển đổi kiểu dữ liệu phân loại sang số nhị phân

Có 4 cột có kiểu dữ liệu object là: StoreType, Assortment, Promolnterval và StateHoliday.

Label encoder: Promolnterval, StateHoliday

Dummy encoding: StoreType, Assortment

- StoreType : 'StoreType_a', 'StoreType_b', 'StoreType_c', 'StoreType_d'.
- Assortment: 'Assortment_a', 'Assortment_b', 'Assortment_c'.

3.3. Tiền xử lý dữ liệu

- Chuyển đổi kiểu dữ liệu phân loại sang số nhị phân

SchoolHoliday	Year	Month	StoreType_a	StoreType_b	StoreType_c	StoreType_d	Assortment_a	Assortment_b	Assortment_c
1	2015	7	0	0	1	0	1	0	0
1	2015	7	0	0	1	0	1	0	0
1	2015	7	0	0	1	0	1	0	0
1	2015	7	0	0	1	0	1	0	0
1	2015	7	0	0	1	0	1	0	0
...
1	2013	1	0	0	0	1	0	0	1
1	2013	1	0	0	0	1	0	0	1
1	2013	1	0	0	0	1	0	0	1
1	2013	1	0	0	0	1	0	0	1
1	2013	1	0	0	0	1	0	0	1

Bảng dữ liệu sau khi dùng label encoder và dummy encoding

3.3. Tiền xử lý dữ liệu



3.3. Tiền xử lý dữ liệu

- Xây dựng model Regression

Chia biến, chia tập train - test và loại bỏ biến tương quan mạnh: Customer, Open

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import
r2_score,mean_squared_error,mean_absolute_error
import math

X_train, X_test, y_train, y_test_open =
train_test_split(combined_data_open.drop(['Sales','Customers','Open'],a
xis=1),
combined_data_open['Sales'], test_size=0.2, random_state=23)
```

3.3. Tiền xử lý dữ liệu

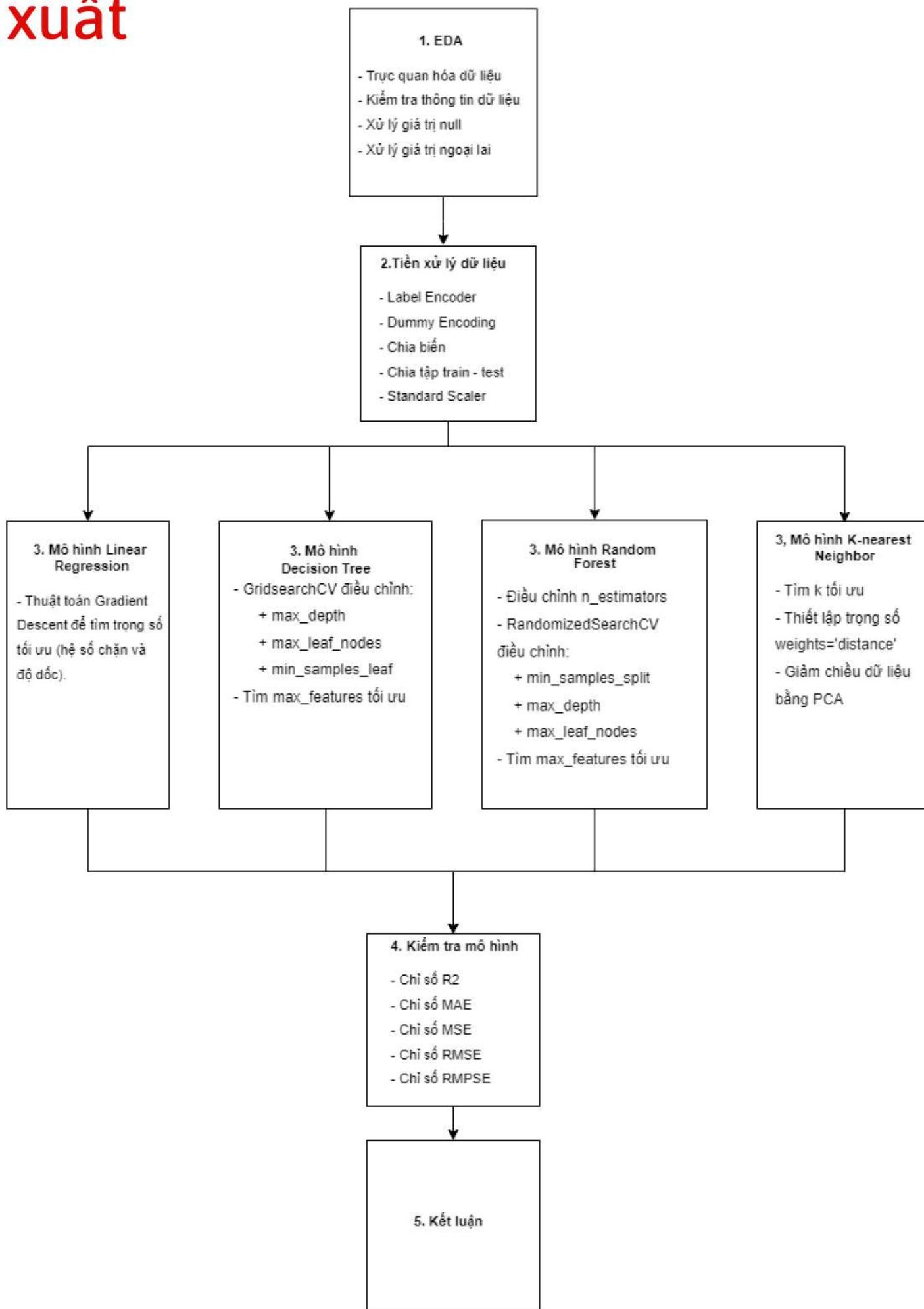
- Xây dựng model Regression

Dùng StandardScaler chia tỷ lệ cho tập huấn luyện và kiểm tra các biến độc lập để giảm kích thước xuống các giá trị nhỏ hơn phù hợp với việc chạy mô hình.

```
array([[-1.02784156, -1.46288917, -0.1750752 , ..., -1.05424542,
       -0.09920987,  1.07508071],
      [-1.25787501,  1.43870354, -0.24599842, ..., -1.05424542,
       -0.09920987,  1.07508071],
      [ 0.21869109, -0.88257063, -0.24937571, ...,  0.94854574,
       -0.09920987, -0.93016273],
      ...,
      [-0.57399176, -0.88257063,  0.24033218, ...,  0.94854574,
       -0.09920987, -0.93016273],
      [-1.62468512, -1.46288917,  1.31768953, ..., -1.05424542,
       -0.09920987,  1.07508071],
      [-0.35639255,  0.858385 ,  2.17889996, ...,  0.94854574,
       -0.09920987, -0.93016273]])
```

Dữ liệu X_train sau khi Standard Scaler

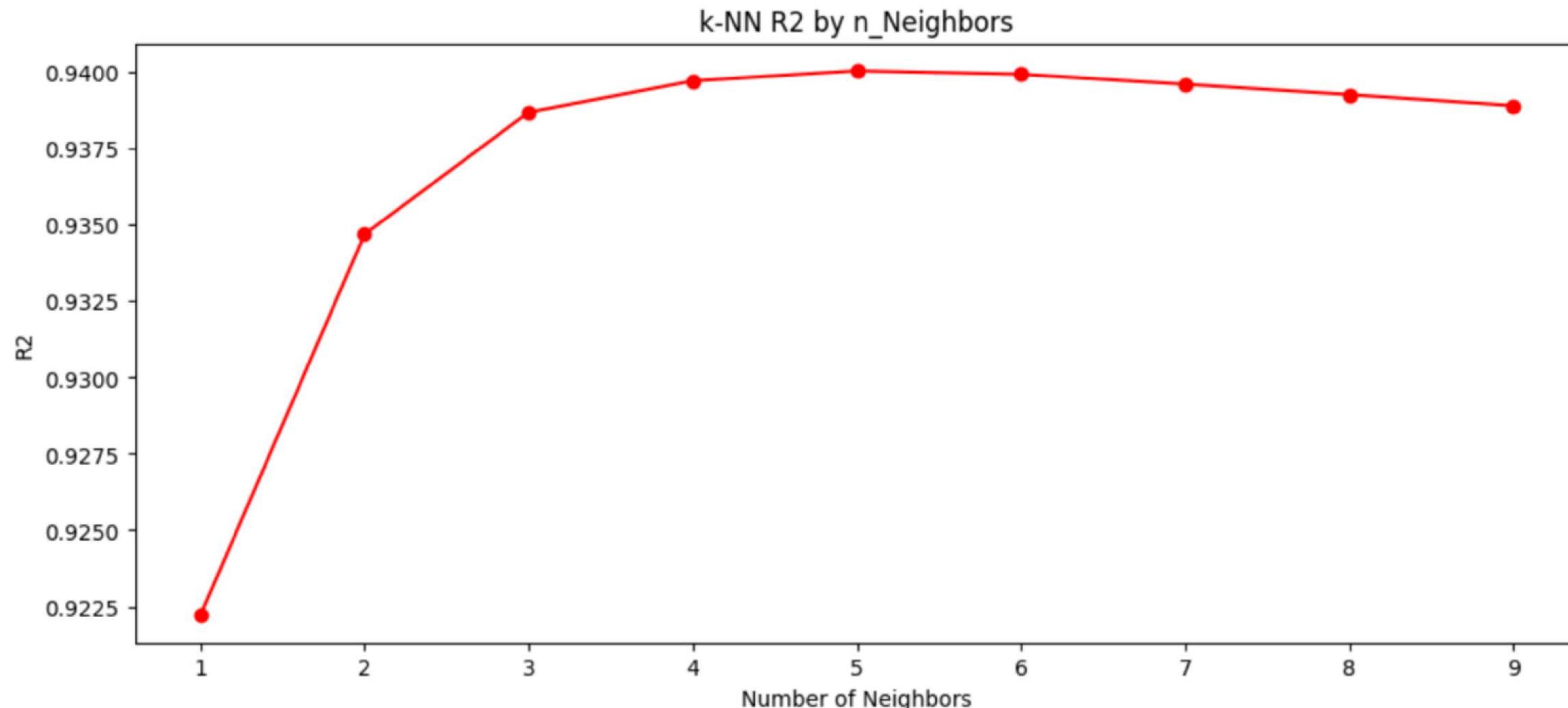
3.4. Mô hình đề xuất



4. KẾT QUẢ THỰC NGHIỆM VÀ ĐỀ XUẤT

4.1.1 Mô hình K-Nearest Neighbor

- Tìm hệ số K tối ưu



4. KẾT QUẢ THỰC NGHIỆM VÀ ĐỀ XUẤT

4.1.1 Mô hình K-Nearest Neighbor

Kết quả đánh giá của mô hình KNN với k=5

R2	0.931
MAE	487.2
MSE	1149970.0
RMSE	1072.4
RMSPE	0.23176

4. KẾT QUẢ THỰC NGHIỆM VÀ ĐỀ XUẤT

4.1.1 Mô hình K-Nearest Neighbor

- Tiến hành đánh trọng số

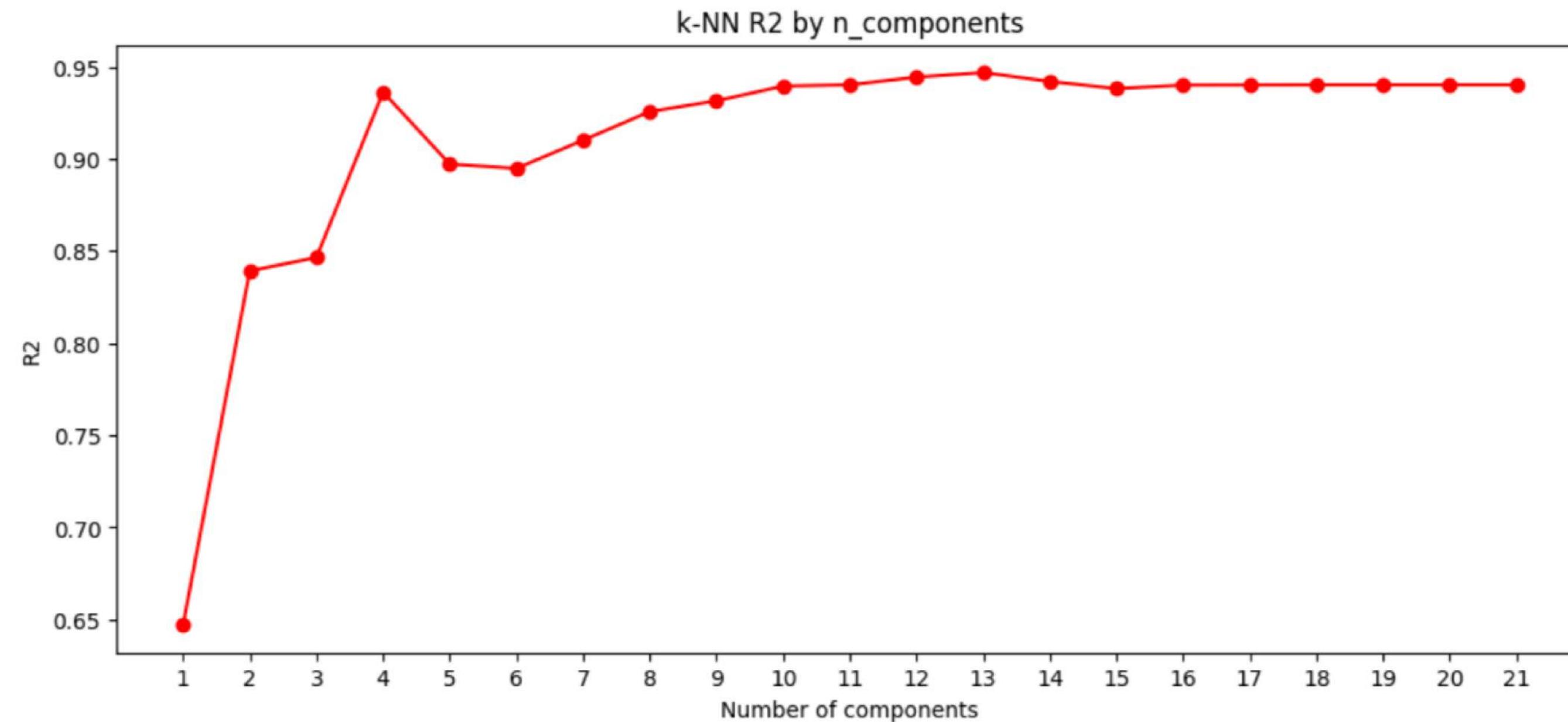
```
KNN_original = KNeighborsRegressor(n_neighbors=5,  
algorithm='auto', leaf_size=30, p=2, metric='minkowski',  
weights='distance')
```

	Mô hình chưa điều chỉnh	Mô hình đánh trọng số
R2	0.931	0.940
MAE	487.2	460.9
MSE	1149970.0	994853.8
RMSE	1072.4	997.4
RMSPE	0.23176	0.23253

4. KẾT QUẢ THỰC NGHIỆM VÀ ĐỀ XUẤT

4.1.1 Mô hình K-Nearest Neighbor

- Thực hiện thay đổi chiều dữ liệu, R2 đạt cao nhất tại n_components =13



4. KẾT QUẢ THỰC NGHIỆM VÀ ĐỀ XUẤT

4.1.1 Mô hình K-Nearest Neighbor

- k = 5
- algorithm='auto'
- leaf_size=30
- p=2, metric='minkowski'
- weights='distance'
- n_components =13

	Mô hình chưa điều chỉnh	Mô hình đã điều chỉnh
R2	0.931	0.947
MAE	487.2	441.7
MSE	1149970.0	883274.0
RMSE	1072.4	939.8
RMSPE	0.23176	0.22318

4. KẾT QUẢ THỰC NGHIỆM VÀ ĐỀ XUẤT

4.1.2 Mô hình Linear Regression

- Mô hình chưa điều chỉnh

R2	0.784
MAE	984.2
MSE	3579884.7
RMSE	1892.0
RMSPE	0.34173

4. KẾT QUẢ THỰC NGHIỆM VÀ ĐỀ XUẤT

4.1.2 Mô hình Linear Regression

- Sử dụng thuật toán Gradient Descent cho mô hình Linear Regression

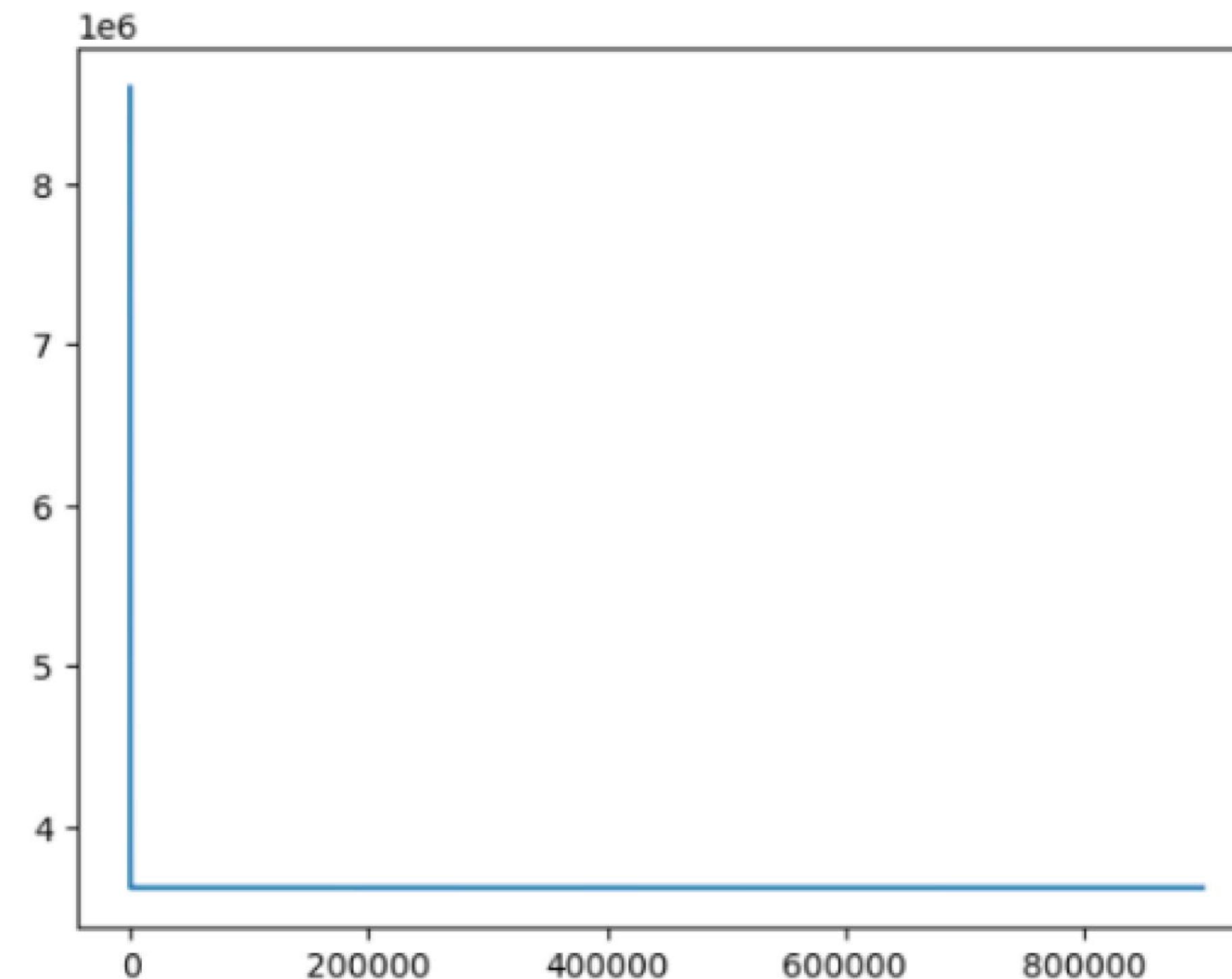
Tính toán các trọng số của mô hình với learning rate = 0.556 và epochs = 200000 (số vòng lặp tối đa)

```
Cost is: 8593986.492160352
Cost is: 3616048.7993090674
Cost is: 3615981.8394388487
Cost is: 3615918.91830362
Cost is: 3615859.7923040404
Cost is: 3615804.2325336444
Cost is: 3615752.0238926336
Cost is: 3615702.9642551155
Cost is: 3615656.863686576
Cost is: 3615613.5437085396
```

Hình: Kết quả giảm dần của hàm chi phí khi chạy thuật toán Gradient Descent

4.1.2 Mô hình Linear Regression

- Sử dụng thuật toán Gradient Descent cho mô hình Linear Regression



4.1.2 Mô hình Linear Regression

- Sử dụng thuật toán Gradient Descent cho mô hình Linear Regression

Hệ số hồi quy:

```
array([ 1.47002157e+01, -1.74732672e+02, -1.07032819e+02, -5.36671200e+00,
       1.14284572e+04,  3.09294501e+02, -1.18801866e+04, -1.60310286e+02,
      -2.53568680e+02,  1.06667811e+03, -5.44722032e+00,  2.92811234e+01,
       1.52342285e+02,  2.47851618e+02, -3.07150989e+01,  6.19889673e+02,
      -4.62800017e+01, -1.12572277e+02, -1.91057595e+02, -2.81769640e+02,
       2.46691340e+02])
```

Hệ số chặn:

```
6936.608734240404
```

4. KẾT QUẢ THỰC NGHIỆM VÀ ĐỀ XUẤT

4.1.2 Mô hình Linear Regression

- Kết quả của mô hình Linear Regression sau khi chạy Gradient Descent

R2	0.784
MAE	984.3
MSE	3580237.9
RMSE	1892.1
RMSPE	0.34174

4.1.3 Mô hình Decision Tree

- Mô hình chưa điều chỉnh

R2 in train	0.964
R2 in test	0.9515
MAE	419.9
MSE	803718.4
RMSE	896.5
RMSPE	0.18057

4.1.3 Mô hình Decision Tree

- Cắt tỉa Decision Tree bằng cách đặt ra các giới hạn cho cây
- GridSearchCV

```
'max_depth': [3000, 4000, 5000, 6000, 7000],  
'min_samples_leaf': [1, 2, 3, 4, 5, 6, 7, 10, 15, 20],  
'max_leaf_nodes' : [10000, 11000, 12000, 13000, 14000, 15000]
```

- Kết quả

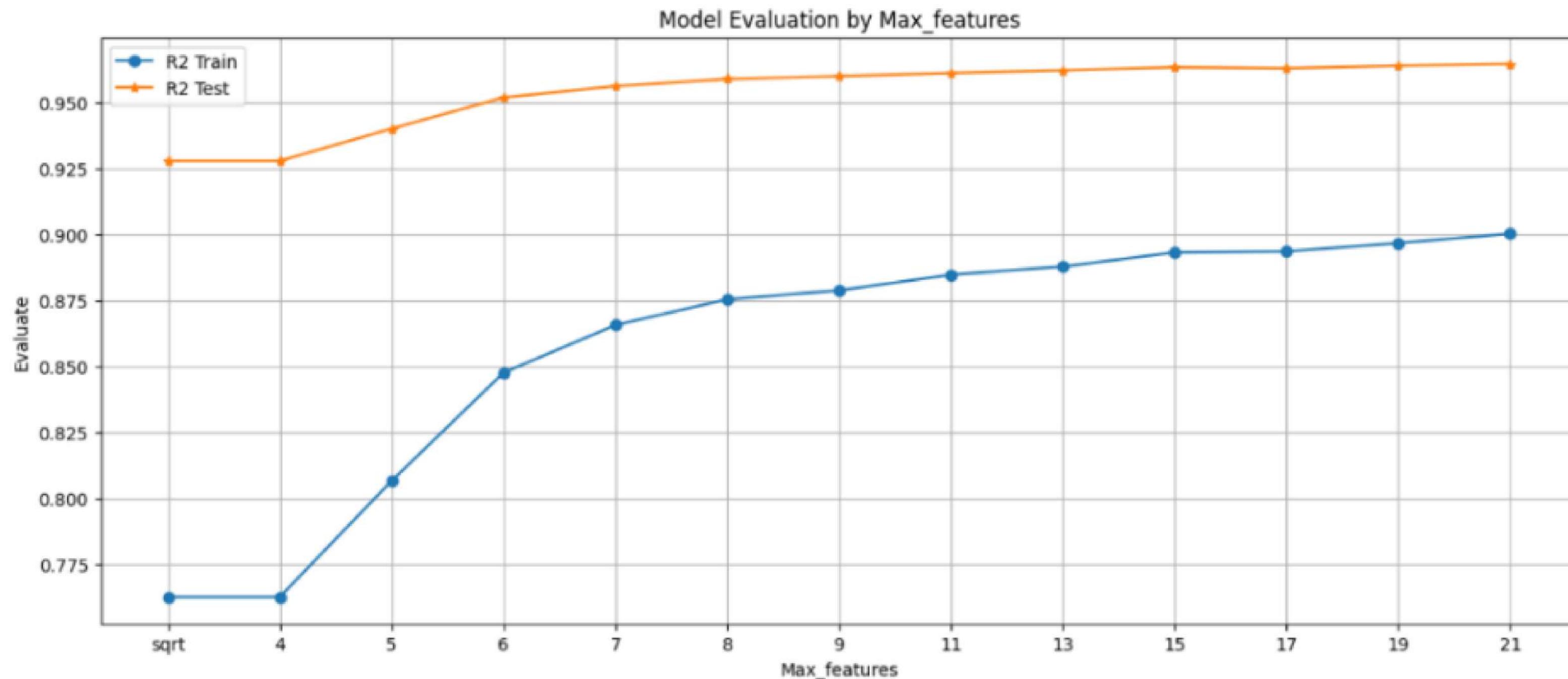
max_depth = 3000

min_samples_leaf = 5

max_leaf_nodes = 15000

4.1.3 Mô hình Decision Tree

- Tìm max_features



Hình: Kiểm tra max_features và hiệu quả của mô hình Decision Tree

4.1.3 Mô hình Decision Tree

- Mô hình đã điều chỉnh

- max_depth = 3000
- min_samples_leaf = 5
- max_leaf_nodes = 15000
- max_feature = 15

	Mô hình chưa điều chỉnh	Mô hình đã điều chỉnh
R2 in train	0.964	0.893
R2 in test	0.9515	0.9632
MAE	419.9	364.0
MSE	803718.4	611153.6
RMSE	896.5	781.8
RMSPE	0.18057	0.11078

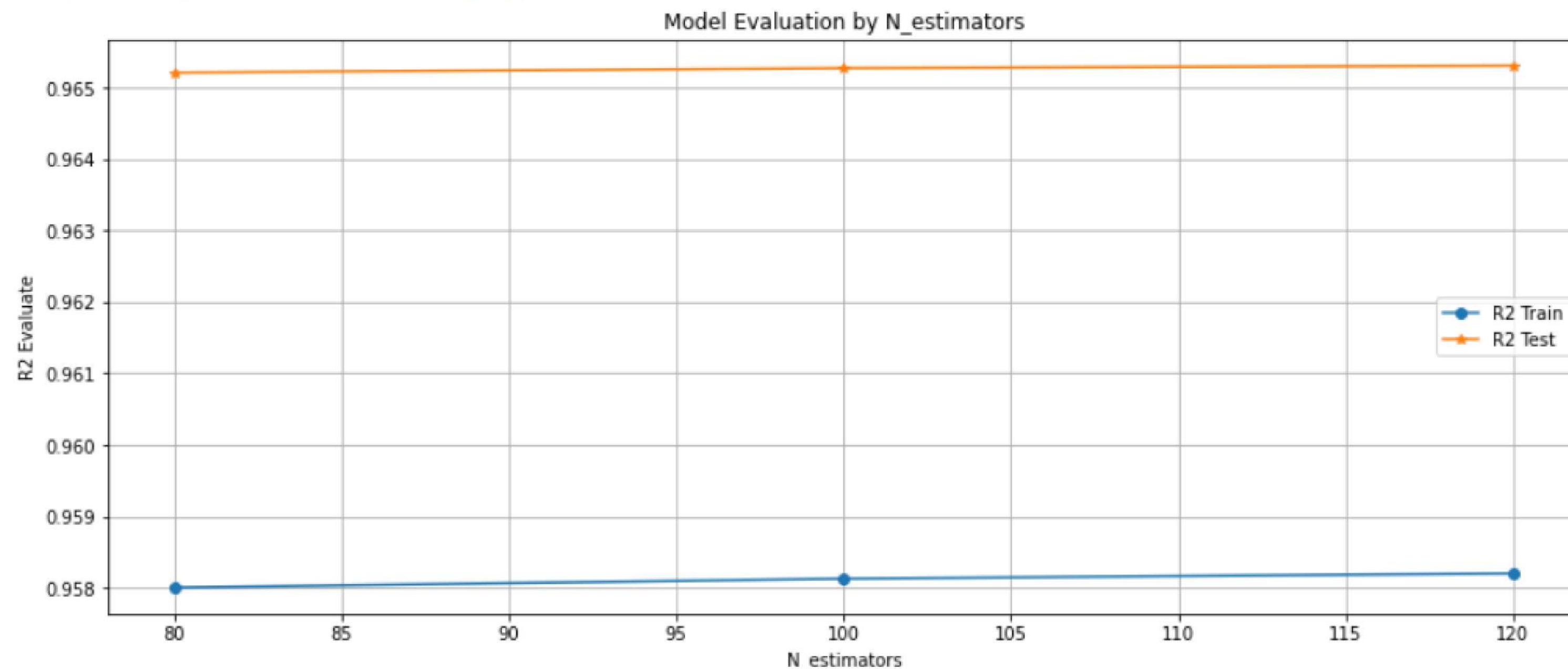
4.1.4 Mô hình Random Forest

- Mô hình chưa điều chỉnh

R2 in train	0.9581
R2 in test	0.9653
MAE	356.7
MSE	576260.7
RMSE	759.1
RMSPE	0.12554

4.1.4 Mô hình Random Forest

- Điều chỉnh `n_estimators` với các thông số ngẫu nhiên tương ứng là 80, 100 và 120



4.1.4 Mô hình Random Forest

- Kết quả mô hình với n_estimators = 120

	Mô hình chưa điều chỉnh	n_estimator = 120
R2 in train	0.9581	0.9653
R2 in test	0.9653	0.9582
MAE	356.7	356.5
MSE	576260.7	575656.4
RMSE	759.1	758.7
RMSPE	0.12554	0.12452

4.1.4 Mô hình Random Forest

- Dùng RandomizedSearchCV điều chỉnh: max_depth, min_samples_split, max_leaf_nodes

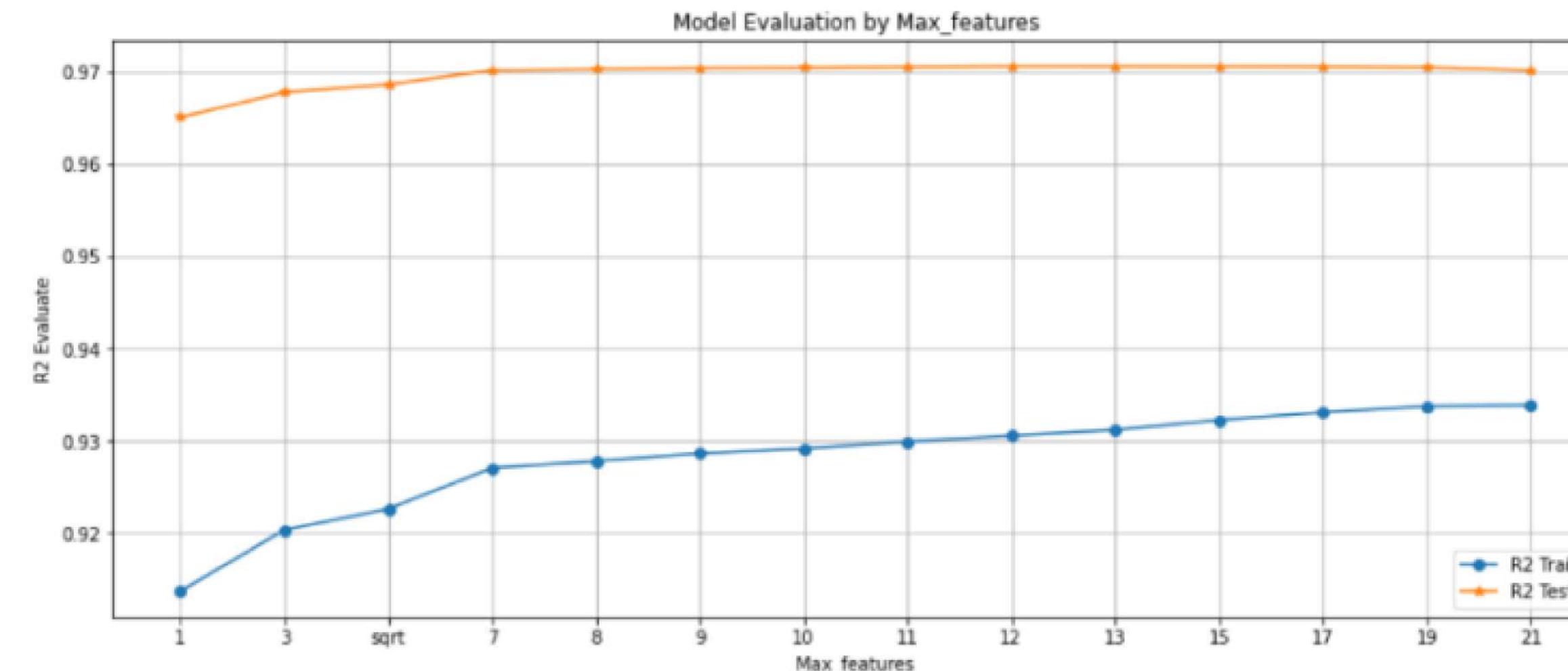
Kết quả chạy thu được là:

- Best estimator: RandomForestRegressor(max_depth=4000, min_samples_split=10, n_estimators=120, random_state=42)
- Best parameter: {'min_samples_split': 10, 'max_leaf_nodes': None, 'max_depth': 4000}
- R2 score: 0.9701706801461882

4.1.4 Mô hình Random Forest

- Điều chỉnh max_features để tìm ra giá trị max_features phù hợp của mô hình

```
features = [1, 3, 'sqrt', 7, 8, 9, 10, 11, 12, 13, 15, 17, 19, 21]
for i in features:
    regressor = RandomForestRegressor(max_depth=4000,
random_state=42, max_leaf_nodes=None, min_samples_split=10,
n_estimators=120, max_features=i)
```



Hình: Kiểm tra max_features trên mô hình Random Forest

4.1.4 Mô hình Random Forest

- Mô hình đã điều chỉnh

- n_estimators=120
- max_depth=4000
- min_samples_split=120
- max_leaf_nodes = None
- max_features: 12

	Mô hình chưa điều chỉnh	Mô hình đã điều chỉnh
R2 in train	0.9581	0.9305
R2 in test	0.9653	0.9706
MAE	356.7	325.1
MSE	576260.7	487562.1
RMSE	759.1	698.3
RMSPE	0.12554	0.10782

4. KẾT QUẢ THỰC NGHIỆM VÀ ĐỀ XUẤT

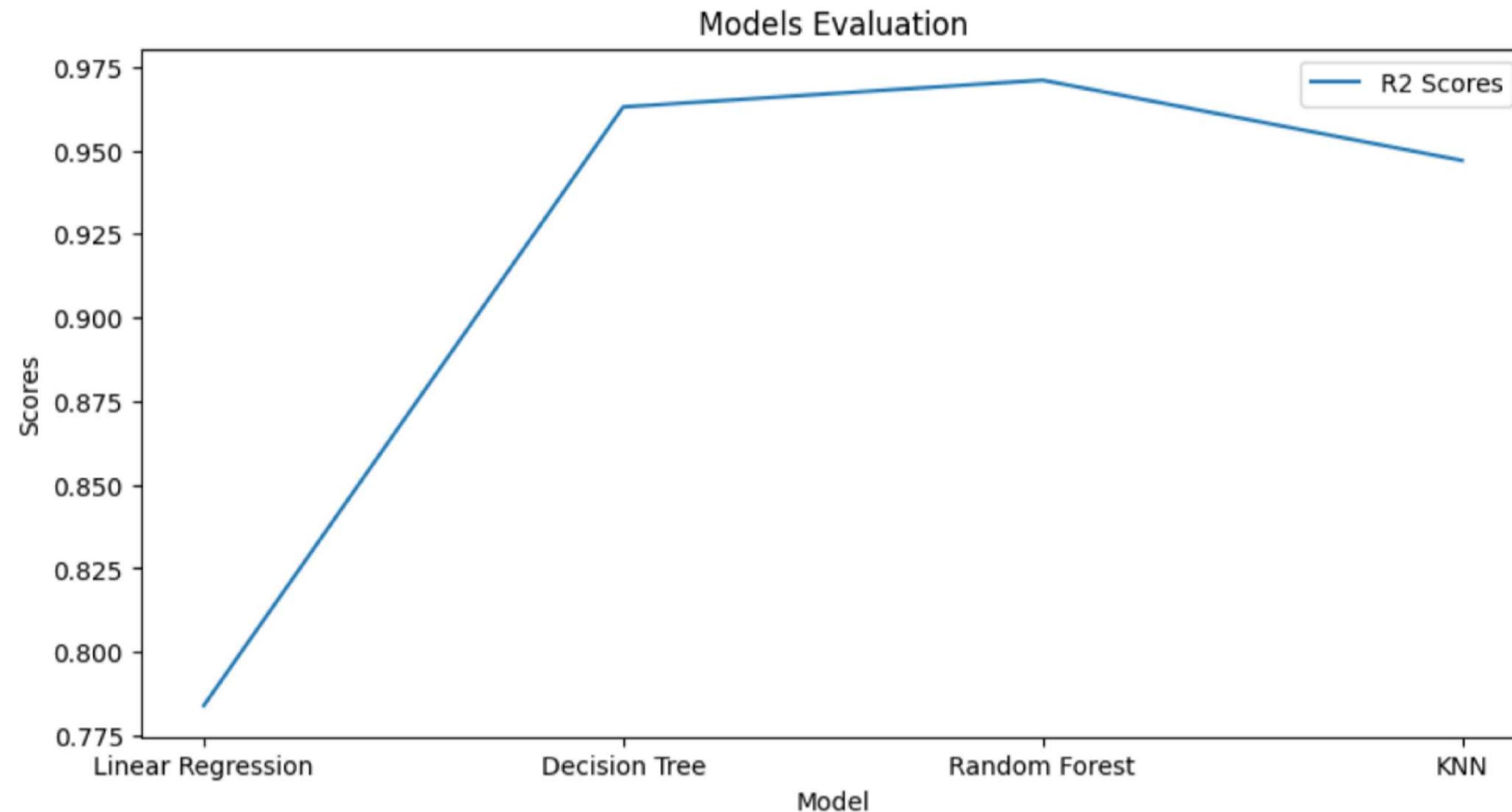
4.2 Báo cáo kết quả

	Model	R2	MAE	MSE	RMSE	RMSPE
0	Linear Regression	0.784	984.4	3580618	1892.2	0.34175
1	Desicion Tree	0.963	365.0	611154	781.8	0.11079
2	Random Forest	0.971	325.1	487562	698.3	0.10782
3	K-Nearest Neighbor	0.947	441.7	883274	939.8	0.22318

Dataframe kết quả các mô hình

4. KẾT QUẢ THỰC NGHIỆM VÀ ĐỀ XUẤT

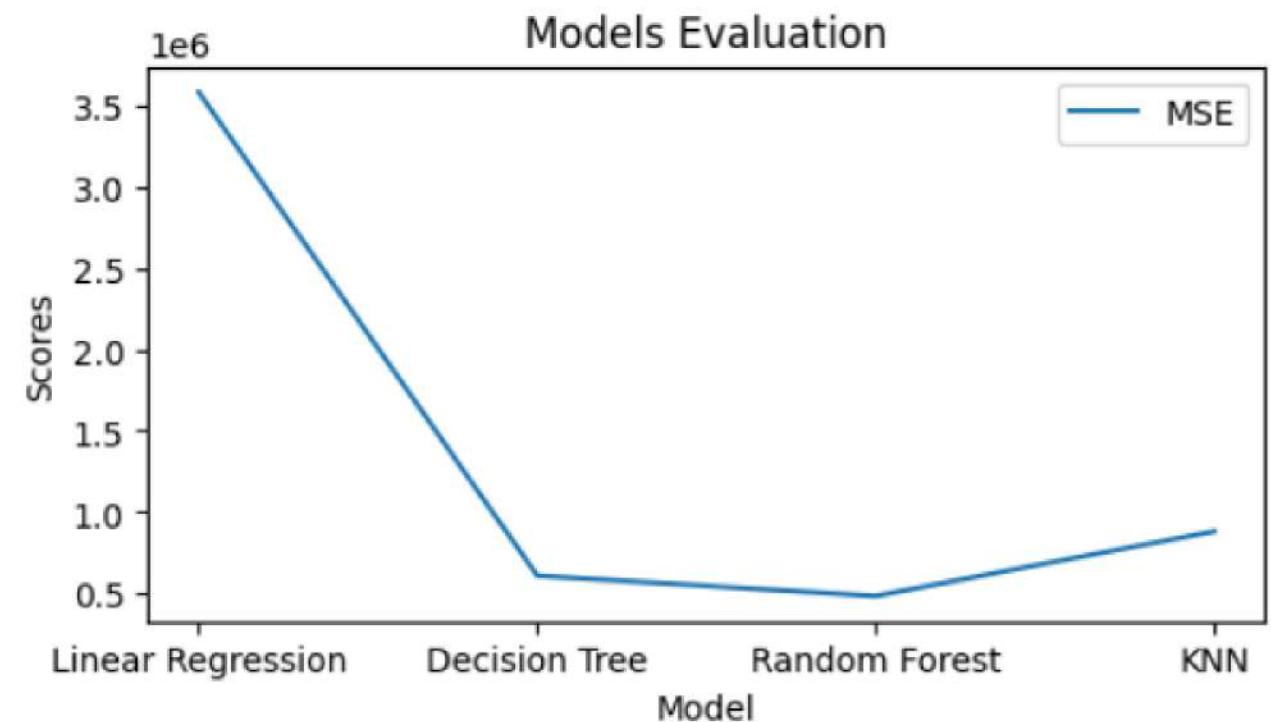
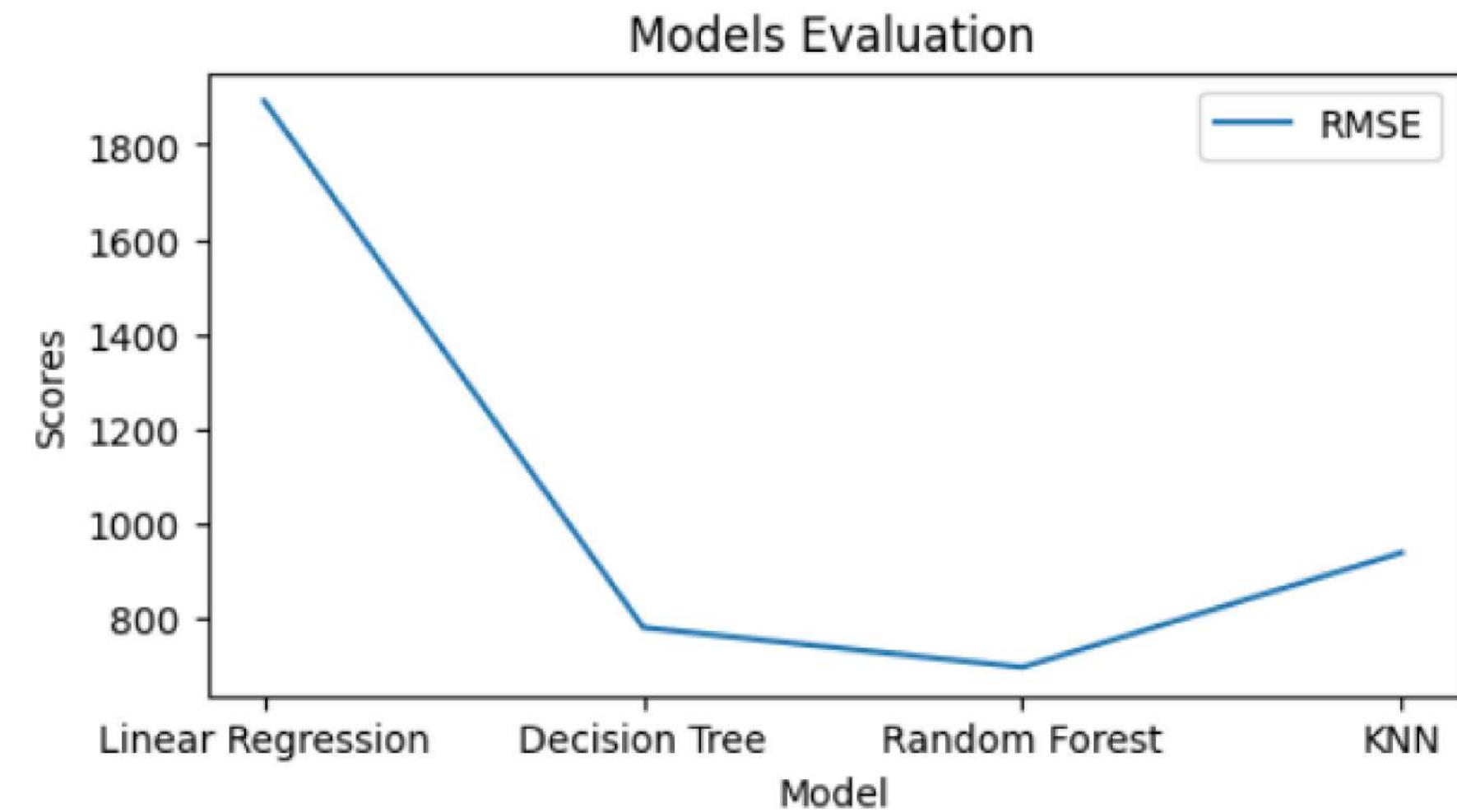
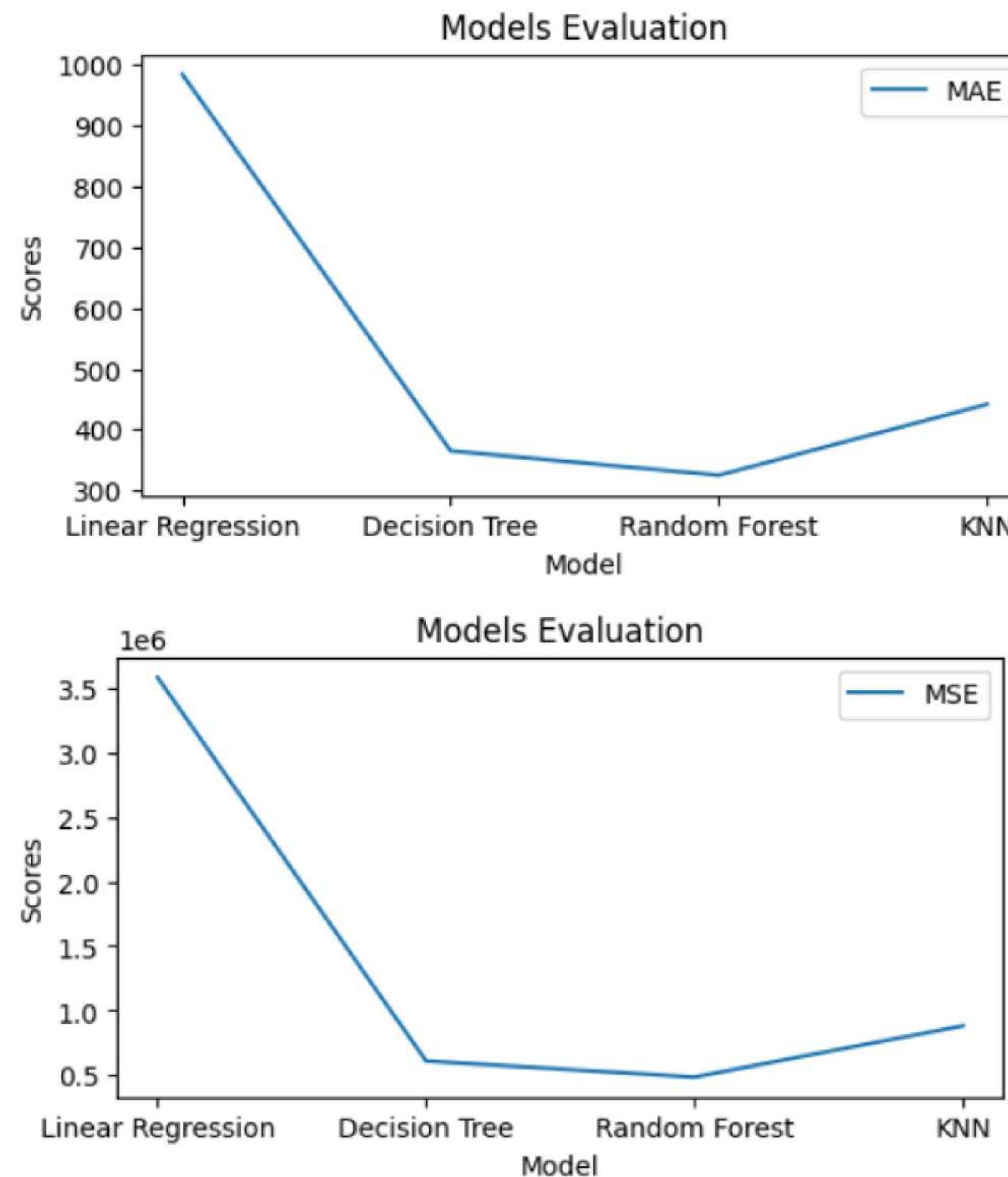
4.2 Báo cáo kết quả



Biểu đồ thể hiện chỉ số R2

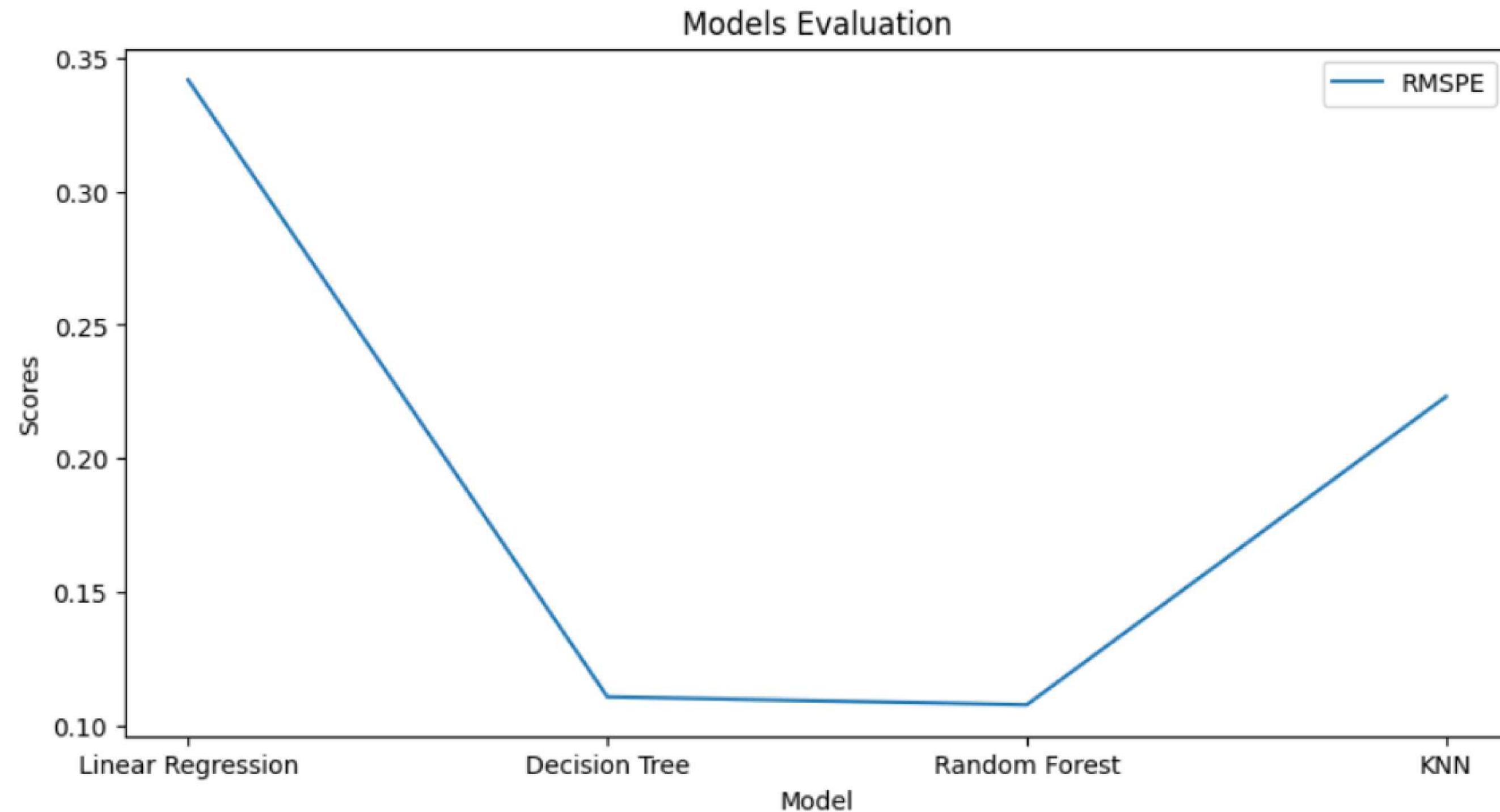
4. KẾT QUẢ THỰC NGHIỆM VÀ ĐỀ XUẤT

4.2 Báo cáo kết quả



4. KẾT QUẢ THỰC NGHIỆM VÀ ĐỀ XUẤT

4.2 Báo cáo kết quả



4. KẾT QUẢ THỰC NGHIỆM VÀ ĐỀ XUẤT

4.2 Báo cáo kết quả

- Kết luận chung: Tất cả 5 chỉ số R2, MAE, MSE, RMSE, RMPSE đều cho ra một kết quả mô hình giống nhau: Random Forest là mô hình tốt nhất, tiếp đến là Decision Tree, KNN và cuối cùng là Linear Regression.

4. KẾT QUẢ THỰC NGHIỆM VÀ ĐỀ XUẤT

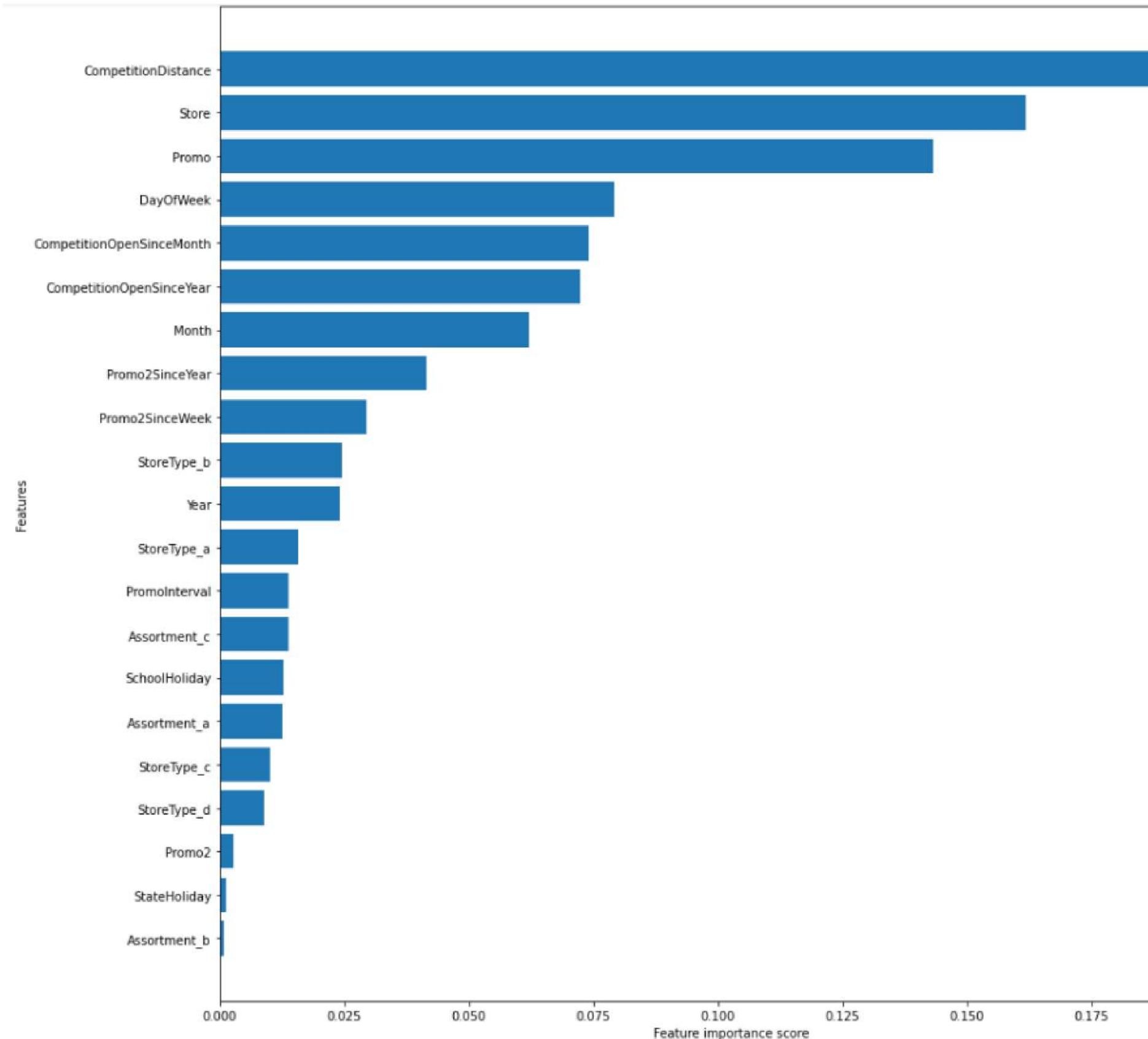
4.2 Báo cáo kết quả

- Features quan trọng ảnh hưởng tới mô hình Decision Tree

	Feature	Importance			
1	CompetitionDistance	0.198299	18	Assortment_a	0.022100
0	Store	0.164238	12	Year	0.022041
9	Promo	0.144933	14	StoreType_a	0.018030
8	DayOfWeek	0.077947	7	PromoInterval	0.013004
3	CompetitionOpenSinceYear	0.074483	11	SchoolHoliday	0.012598
2	CompetitionOpenSinceMonth	0.071056	17	StoreType_d	0.008711
13	Month	0.058147	16	StoreType_c	0.008279
6	Promo2SinceYear	0.040562	20	Assortment_c	0.005264
5	Promo2SinceWeek	0.030060	4	Promo2	0.002446
15	StoreType_b	0.026183	10	StateHoliday	0.001208
			19	Assortment_b	0.000411

4.2 Báo cáo kết quả

- Features quan trọng ảnh hưởng tới mô hình Random Forest



Kết luận chung: 10 features quan trọng nhất tác động đến hiệu quả mô hình là:

- CompetitionDistance
- Store
- Promo
- DayOfWeek
- CompetitionOpenSinceMonth
- CompetitionOpenSinceYear
- Month
- Promo2SinceYear
- Promo2SinceWeek
- StoreType_b

4.3 Đề xuất mô hình áp dụng

- Sau quá trình đào tạo và đánh giá từng mô hình, nhóm nhận thấy mô hình **Random Forest** đạt hiệu quả tốt nhất.
- Vì vậy nhà thuốc Rossmann nên áp dụng mô hình Random Forest để dự đoán doanh thu các cửa hàng nhằm có những điều chỉnh thích hợp, kịp thời trên kế hoạch kinh doanh của mình.

R2 in train	0.9305
R2 in test	0.9706
MAE	325.1
MSE	487562.1
RMSE	698.3
RMSPE	0.10782

5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết luận

- Năm thuộc tính của bộ dữ liệu có ảnh hưởng mạnh mẽ nhất đến doanh thu xếp từ cao đến thấp: CompetitionDistance, Store, Promo, DayOfWeek, CompetitionOpenSinceMonth.
- Random Forest > Decision Tree > K-Nearest Neighbors > Linear Regression
- Mô hình Random Forest cho ra chỉ số RMPSE nhỏ hơn bài nghiên cứu trước đó

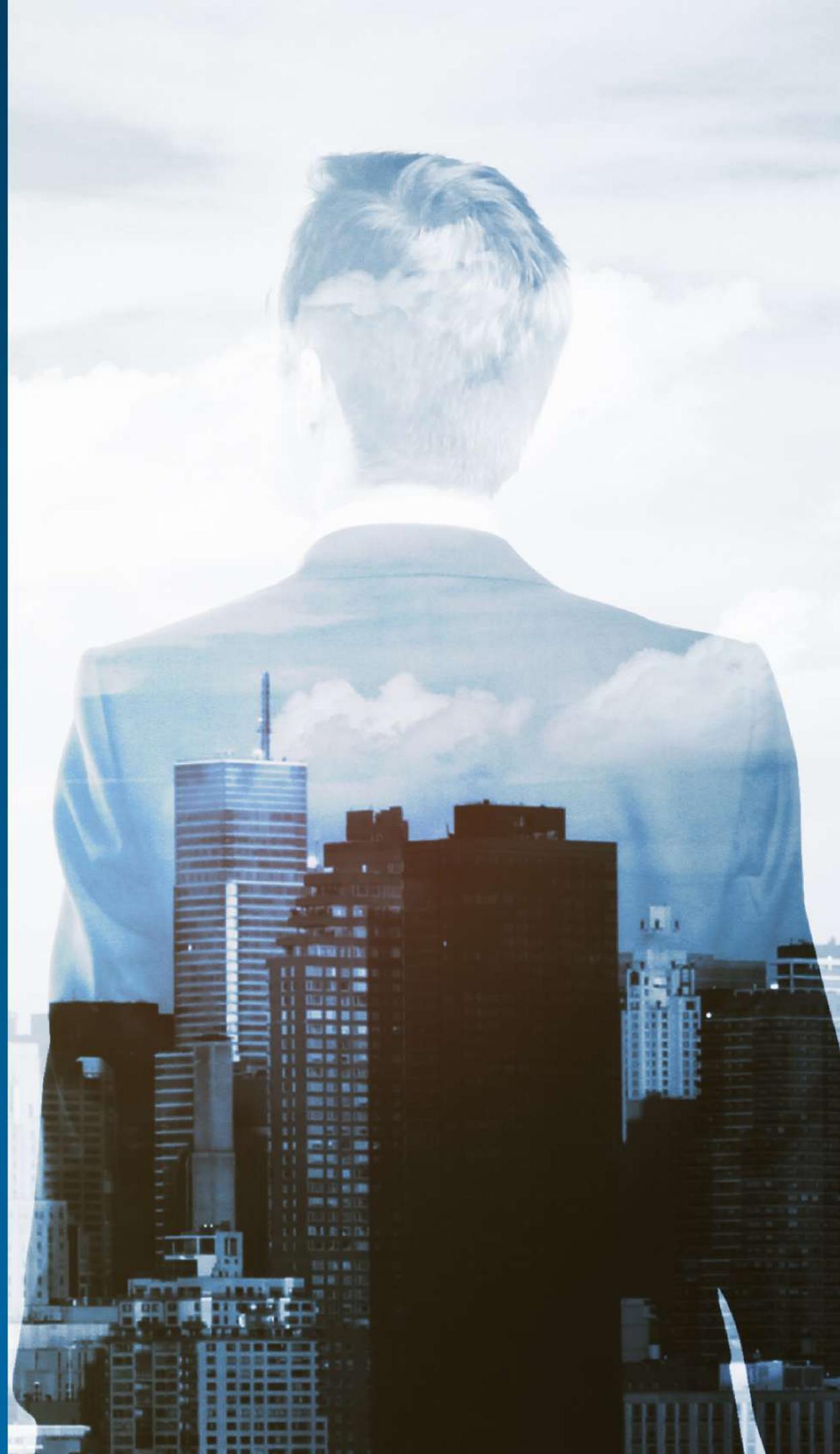
GO



5.2. Hạn chế

- Hạn chế về thời gian và tài nguyên máy
- Nền tảng hỗ trợ việc chạy mô hình chưa phù hợp
- Tập dữ liệu lớn chưa được cập nhật
- Nhiều siêu tham số được tìm hiểu nhưng chưa áp dụng vào bài
- Tập trung vào mô hình học máy, chưa mở rộng so sánh mô hình khác

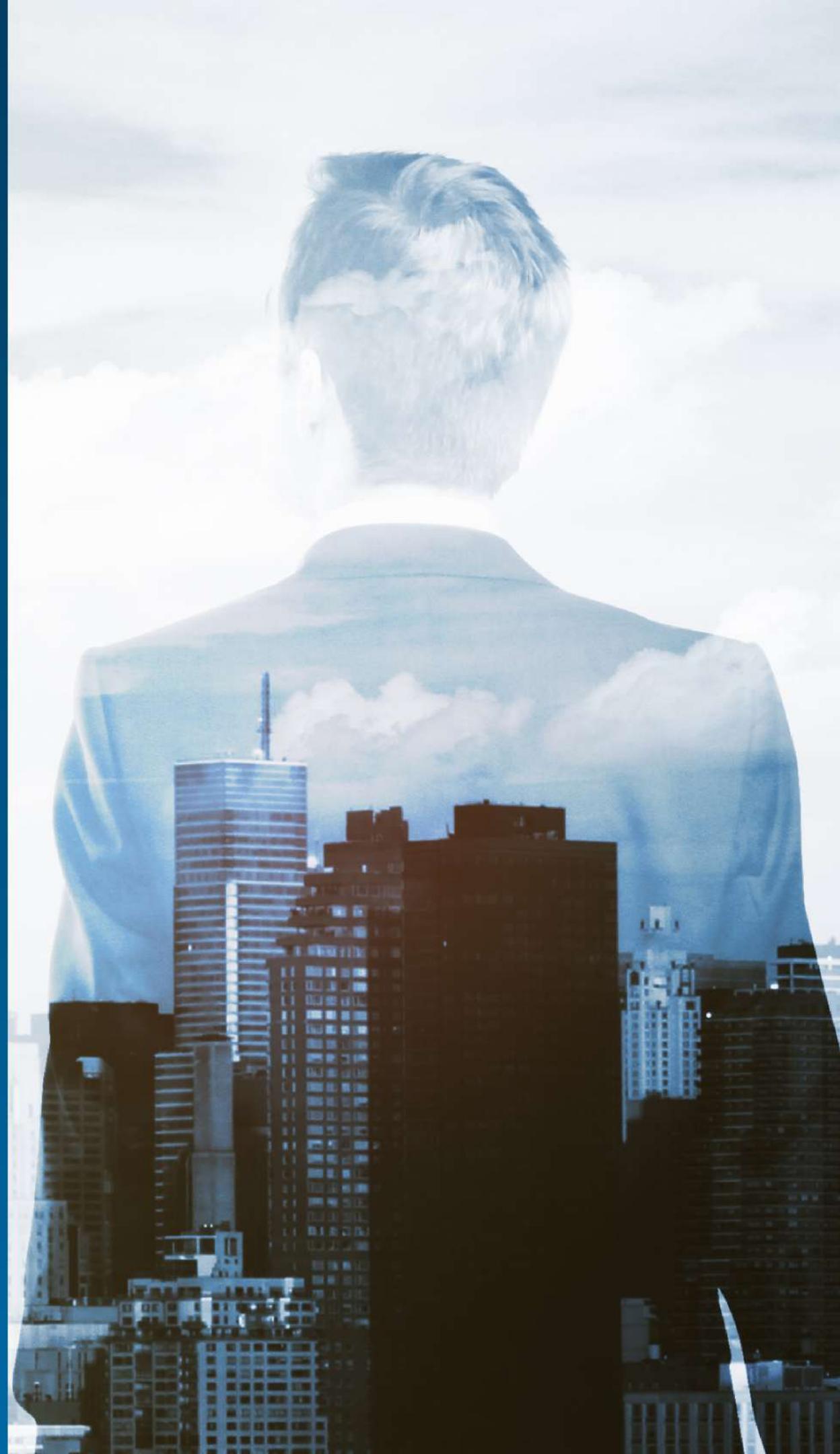
GO



5.2. Hướng phát triển

- Mở rộng so sánh hiệu quả giữa các loại mô hình và đề xuất so sánh hiệu quả giữa phân tích thời gian và học máy
- Ứng dụng thêm các giải pháp tối ưu hiệu suất mô hình hiệu quả hơn
- Mở rộng thử nghiệm nhiều siêu tham số khác nhau

GO



THANK YOU

Vì đã lắng nghe bài thuyết trình.

```
    /ime + 1
    copy
    .SaveDialog()
    os.path.split(filePath)
    ch + "W"
    ertainment
    /ext = "Obj Sequence will be saved as:\n\n"
    /ath + objName + "###.obj\n\n"
    name = str(fromTime) + " to " + str(toTime) + " for " + str(animLength,
    A = c4d.gui.QuestionDialog(questionDialogText)

needBool = True:

loop through animation and export frames
for x in range(0,animLength):

    change frame, redraw view
    time = c4d.BaseTime(fromTime,docFps) + c4d.BaseTime(x,docFps)
    tTime(moveTime)
    ntAdd(c4d.EVENT_FORCEREDRAW)
    lView(c4d.DRAWFLAGS_FORCEFULLREDRAW)

    ar
    Text("Exporting " + str(x) + " of " + str(animLength))

    (doc.GetTime()).GetFrame(doc-->
```

