

Dự đoán điểm tâm trạng (Mood Score) bằng Machine Learning

Nhóm 9:

- Nguyễn Ngọc Hải Hau-2022605947
- Nguyễn Thành Công Hau-2022606702
- Vũ Minh Đức 11221425
- Nguyễn Hoàng Nguyên 11224818
- Nguyễn Trọng Vỹ 11227025





Giới thiệu bài toán

Nguồn dữ liệu: <https://www.kaggle.com/datasets/abhishek9/digital-habits-vs-mental-health-dataset/data>

Dự đoán mức độ "mood" (điểm tâm trạng) của người dùng dựa trên:

- screen_time_hours – Thời gian sử dụng thiết bị màn hình (giờ/ngày)
- social_media_platforms_used – Số lượng nền tảng mạng xã hội sử dụng
- hours_on_TikTok – Số giờ sử dụng TikTok mỗi ngày
- sleep_hours – Thời gian ngủ mỗi đêm (giờ)
- stress_level – Mức độ căng thẳng (thang điểm, ví dụ: 1–10)

Mục tiêu: Dự đoán biến liên tục mood_score – Điểm tâm trạng → Hồi quy

Dữ liệu có 100000 dòng dữ liệu không có giá trị null và đều là biến numerical

Kết quả mong đợi tìm ra model phù hợp với bài toán 1 trong 5 model: dựa vào r2_score và mse

Các mô hình

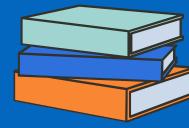
- Linear_Regression
- GradientBoostingRegressor
- RandomForestRegressor
- neural_network
- Polynomial_Regression



Import thư viện

Import trong code	Gói cần cài (pip install ...)	Ghi chú
pandas	pandas	Phân tích dữ liệu dạng bảng
numpy	numpy	Tính toán số học
seaborn	seaborn	Trực quan hóa (biểu đồ đẹp hơn matplotlib)
matplotlib.pyplot	matplotlib	Trực quan hóa
sklearn.model_selection	scikit-learn	Các công cụ train/test split, GridSearchCV
lazypredict.Supervised	lazypredict	Chạy nhiều mô hình cùng lúc để so sánh
sklearn.preprocessing	scikit-learn	Chuẩn hóa (StandardScaler, PolynomialFeatures)
sklearn.ensemble	scikit-learn	RandomForestRegressor
sklearn.neural_network	scikit-learn	MLPRegressor
sklearn.linear_model	scikit-learn	LinearRegression
sklearn.metrics	scikit-learn	Đánh giá mô hình (MSE, R ²)





Khám phá dữ liệu

Tổng quan dữ liệu

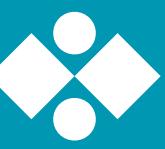
Data columns (total 6 columns):

#	Column	Non-Null Count	Dtype
0	screen_time_hours	100000 non-null	float64
1	social_media_platforms_used	100000 non-null	int64
2	hours_on_TikTok	100000 non-null	float64
3	sleep_hours	100000 non-null	float64
4	stress_level	100000 non-null	int64
5	mood_score	100000 non-null	int64

dtypes: float64(3), int64(3)
memory usage: 4.6 MB

Một số thông số của dữ liệu

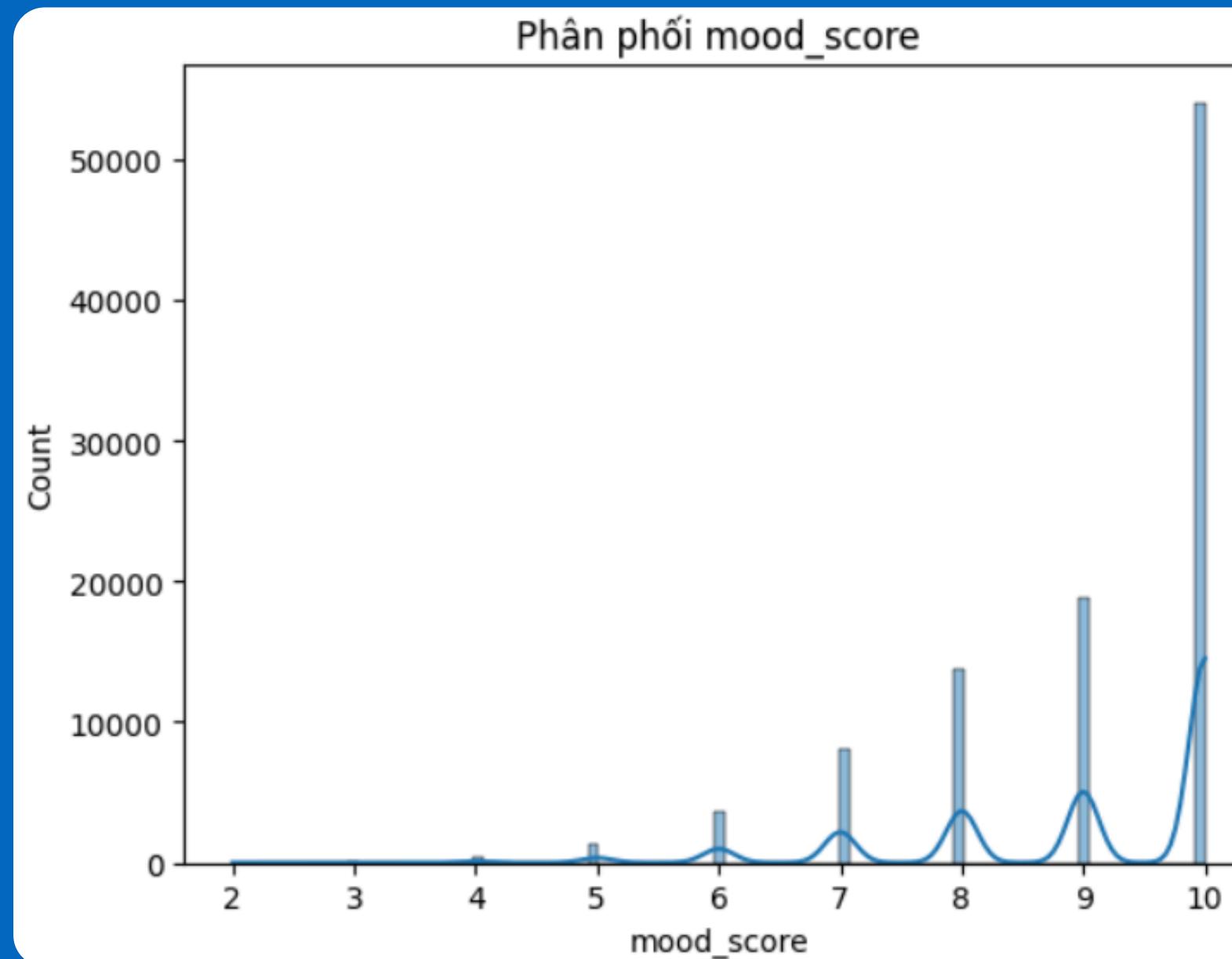
	screen_time_hours	social_media_platforms_used	hours_on_TikTok	sleep_hours	stress_level	mood_score
count	100000.00	100000.00	100000.00	100000.00	100000.00	100000.00
mean	6.00	3.00	2.40	6.99	6.18	9.06
std	1.99	1.41	1.08	1.47	2.05	1.28
min	1.00	1.00	0.20	3.00	1.00	2.00
25%	4.70	2.00	1.60	6.00	5.00	8.00
50%	6.00	3.00	2.30	7.00	6.00	10.00
75%	7.30	4.00	3.10	8.00	8.00	10.00
max	12.00	5.00	7.20	10.00	10.00	10.00



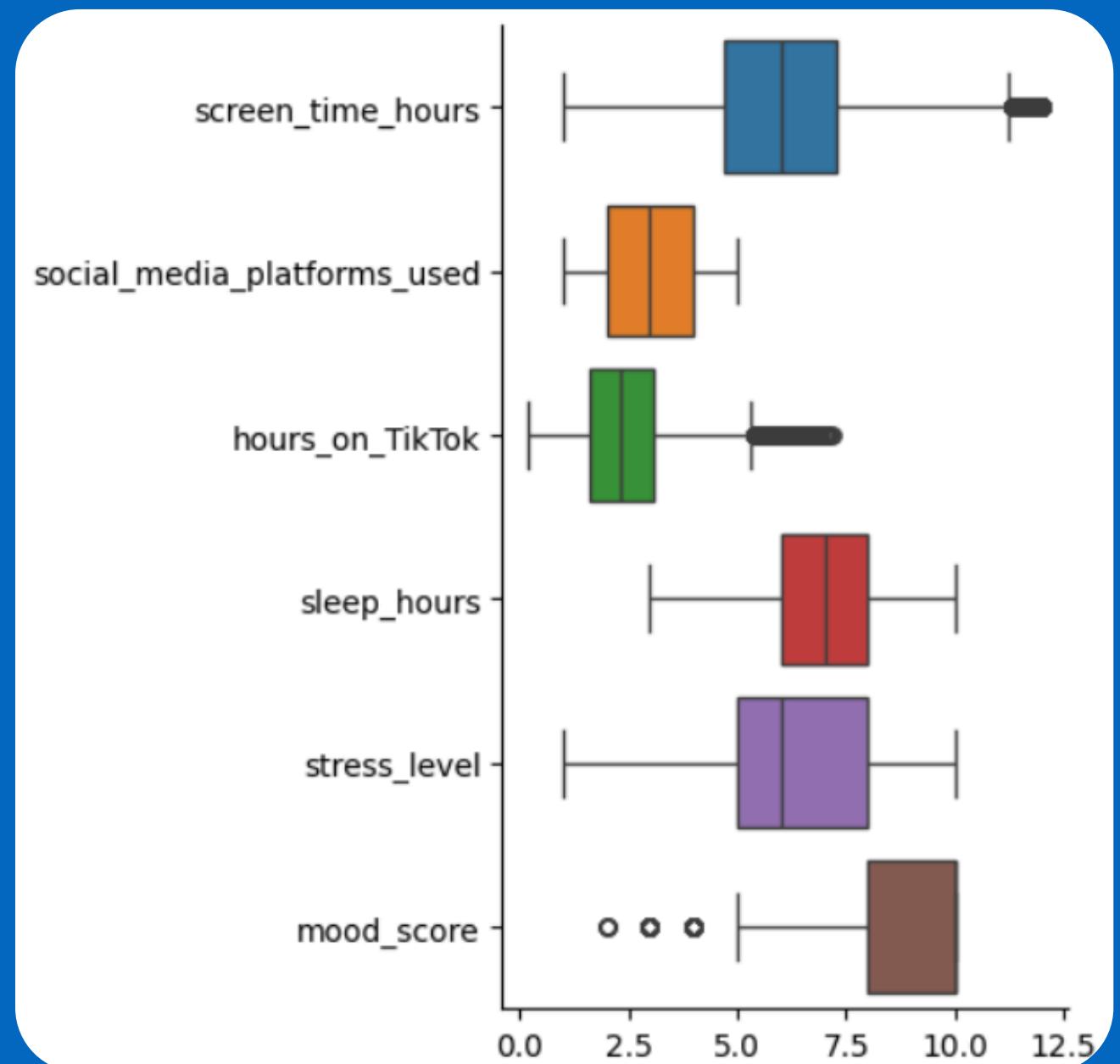


Trực quan hóa dữ liệu

Phân bố các biến mục tiêu



kiểm tra outlier của các feature

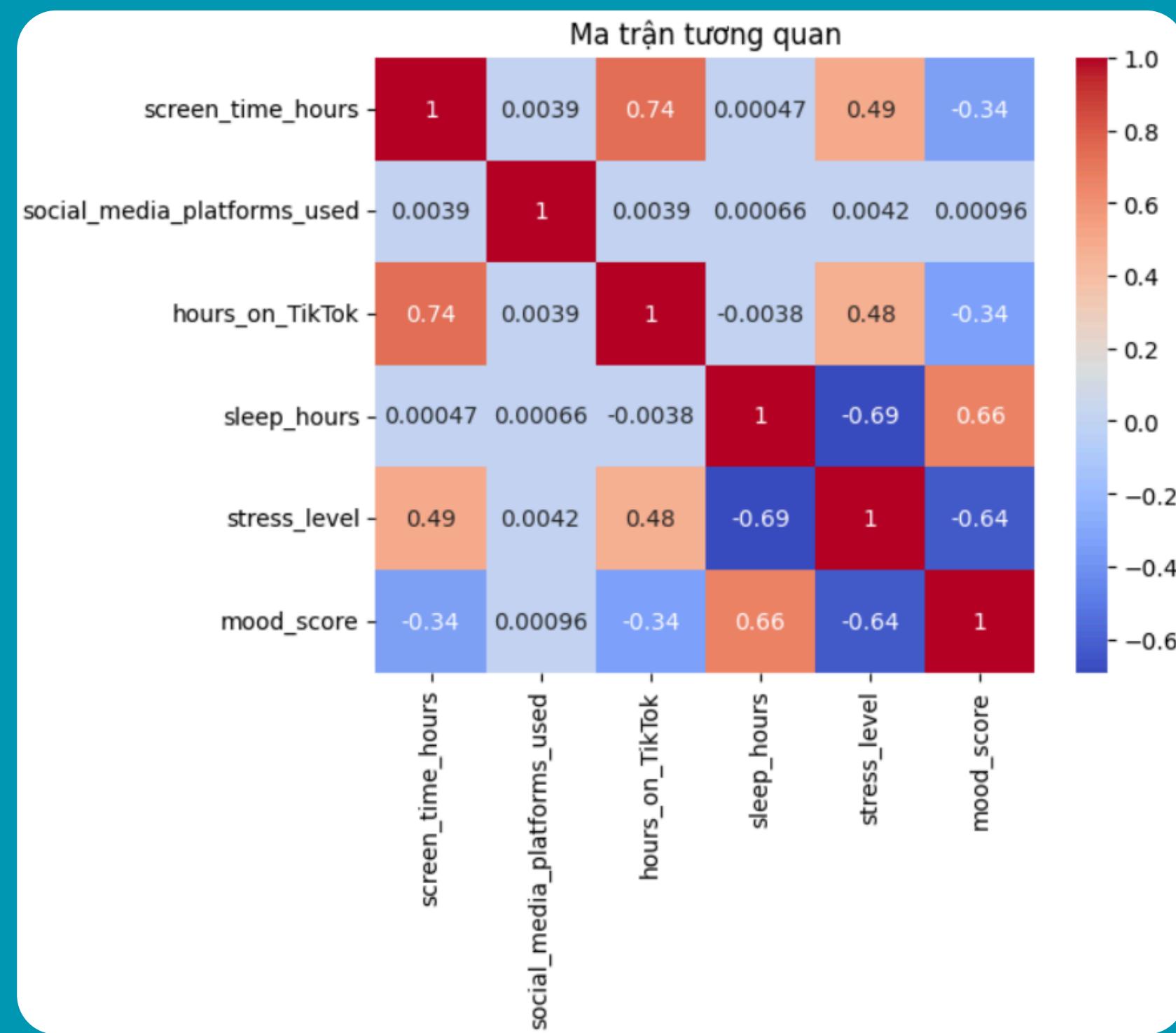




Trực quan hóa dữ liệu



Mối quan hệ các biến





Mô hình RandomForestRegressor (Nguyễn Ngọc Hải)

Lựa chọn model

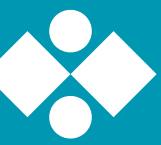
Model	Adjusted R-Squared	R-Squared	RMSE	Time Taken
SVR	0.65	0.66	0.76	0.07
NuSVR	0.65	0.66	0.76	0.53
GradientBoostingRegressor	0.57	0.58	0.84	0.18
RandomForestRegressor	0.55	0.56	0.86	0.37
OrthogonalMatchingPursuitCV	0.54	0.55	0.87	0.39
SGDRegressor	0.54	0.55	0.87	0.31
LassoLarsCV	0.54	0.55	0.87	0.03
LassoLarsIC	0.54	0.55	0.87	0.02
LassoCV	0.54	0.55	0.87	0.10
ElasticNetCV	0.54	0.55	0.87	0.11
BayesianRidge	0.54	0.55	0.87	0.02
Ridge	0.54	0.55	0.87	0.02
RidgeCV	0.54	0.55	0.87	0.01
LinearRegression	0.54	0.55	0.87	0.01
TransformedTargetRegressor	0.54	0.55	0.87	0.02
LinearSVR	0.54	0.55	0.87	0.04
HuberRegressor	0.54	0.55	0.87	0.05
LarsCV	0.53	0.55	0.88	0.05
Lars	0.53	0.54	0.88	0.09
KNeighborsRegressor	0.53	0.54	0.88	0.17
BaggingRegressor	0.51	0.52	0.90	0.07
ExtraTreesRegressor	0.51	0.52	0.90	0.30
PoissonRegressor	0.50	0.51	0.91	0.42

Dựa vào bảng trên có các model phù hợp với dữ liệu bài toán ta chọn RandomForestRegressor vì:

- SVR,NuSVR có R-Squared,RMSE đều tốt hơn RandomForestRegressor nhưng tốc độ huấn luyện lâu và trước khi huấn luyện cũng cần tinh chỉnh dữ liệu khác nhiều.
- GradientBoostingRegressor thì cần xử lý dữ liệu khá chi tiết mới được đưa vào model để huấn luyện có thể dẫn đến overfitting mà R-Squared,RMSE cũng chỉ hơn có 0.02 không đáng kể

=> chọn RandomForestRegressor :

- Nó không yêu cầu chuẩn hóa hay scale dữ liệu.
- Có thể xử lý tốt cả biến liên tục (float) lẫn rời rạc (int).
- Có khả năng tự động nắm bắt quan hệ phi tuyến và tương tác giữa các biến.



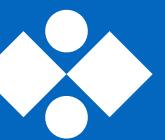
Mô hình RandomForestRegressor (Nguyễn Ngọc Hải)

Lựa chọn tham số model

```
param_grid = {  
    'n_estimators': [50, 100, 150, 200],  
    'max_depth': [10, 15, 20],  
    'min_samples_split': [2, 5],  
    'min_samples_leaf': [1, 3]  
}  
rf = RandomForestRegressor(random_state=42, n_jobs=-1)  
grid_search = GridSearchCV(  
    estimator=rf,  
    param_grid=param_grid,  
    cv=5,  
    scoring='neg_root_mean_squared_error',  
    verbose=2,  
    n_jobs=-1  
)  
grid_search.fit(x, y)  
print("Best Parameters:")  
print(grid_search.best_params_)  
print(f"Best RMSE: {-grid_search.best_score_:.4f}")  
  
Fitting 5 folds for each of 48 candidates, totalling 240 fits  
Best Parameters:  
{'max_depth': 20, 'min_samples_leaf': 3, 'min_samples_split': 2, 'n_estimators': 100}  
Best RMSE: 0.8109
```

Sau khi dùng GridSearchCV để có thể biết được huấn luyện với bài toán trên dùng tham số nào hợp lý cho model RandomForestRegressor thì ta thu được

- RMSE = 0.8109-Nghĩa là sai số trung bình của dự đoán lệch khoảng ~0.81 điểm mood_score trên thang điểm 10. Đây là kết quả khá tốt nếu như dữ liệu bạn có độ nhiễu cao.
- max_depth=20-Mô hình khá sâu, cho phép học được nhiều quan hệ phi tuyến, nhưng vẫn được kiểm soát để tránh overfitting.
- min_samples_split=5, min_samples_leaf=3-Giúp cây không quá rẽ nhánh nhỏ → tránh overfit → tăng khả năng tổng quát hóa.
- n_estimators=100-100 cây là mức hợp lý, cho độ chính xác tốt và thời gian huấn luyện chấp nhận được.

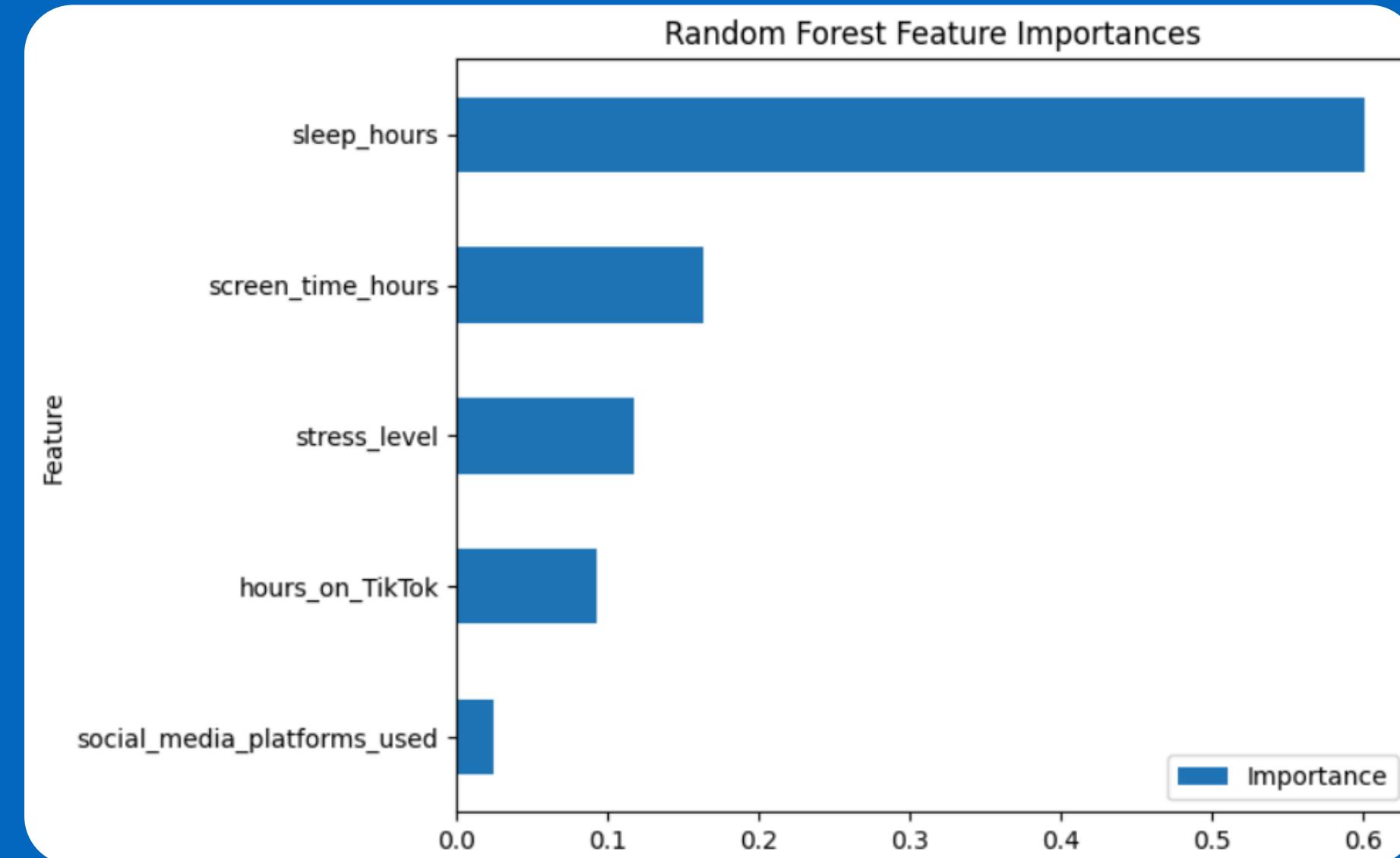


Mô hình RandomForestRegressor (Nguyễn Ngọc Hải)

Huấn luyện model

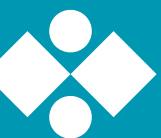
```
model = RandomForestRegressor(  
    max_depth=20,  
    min_samples_leaf=3,  
    min_samples_split=5,  
    n_estimators=100,  
    random_state=42,  
    n_jobs=-1  
)  
model.fit(X_train, y_train)  
y_pred = model.predict(X_test)
```

Phân tích chỉ số ảnh hưởng đến model



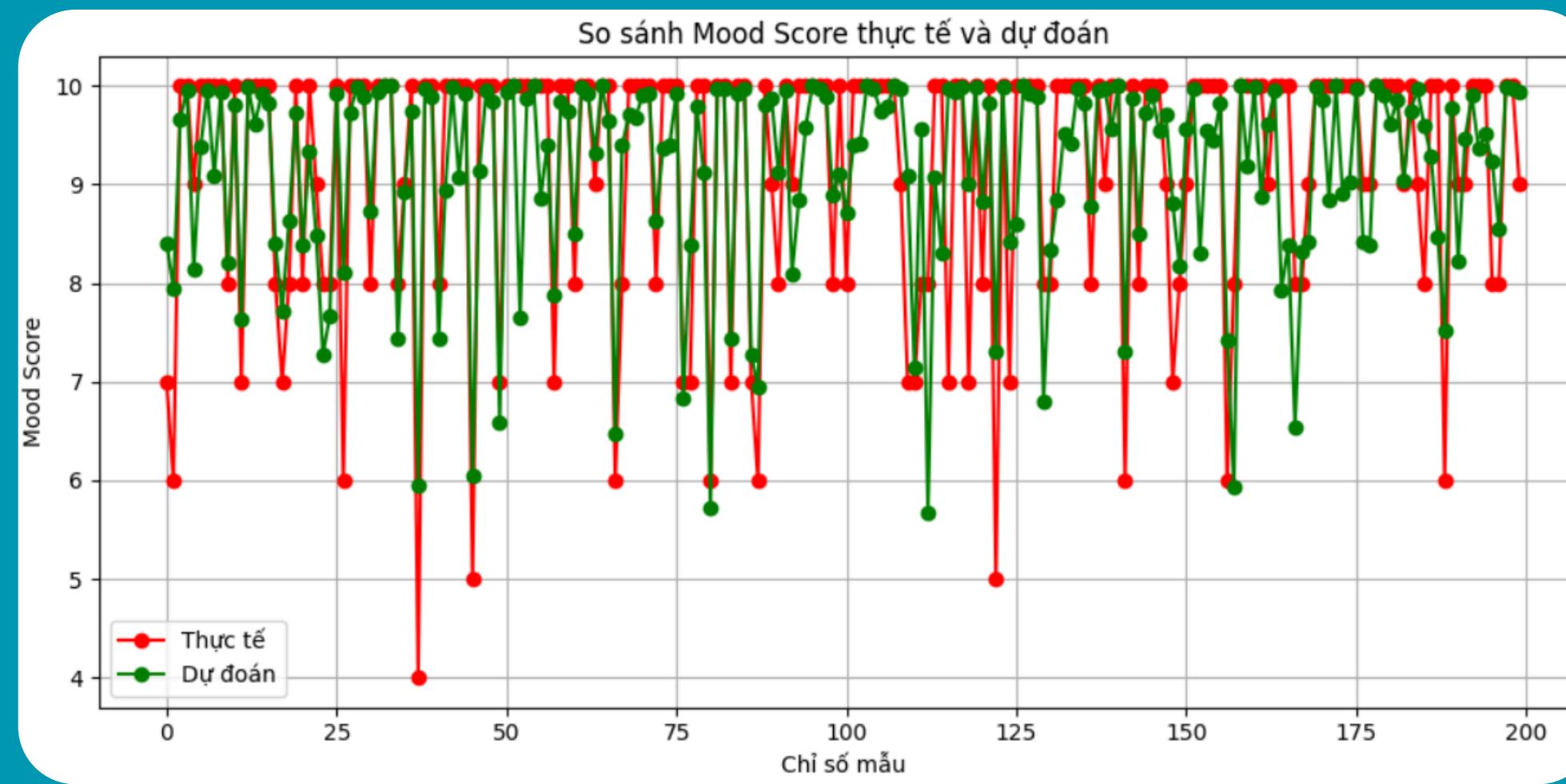
Sau khi huấn luyện model xong ta có thể thấy

- Thời gian ngủ có ảnh hưởng lớn nhất đến mood. Có thể do giấc ngủ ảnh hưởng trực tiếp đến tinh thần và năng lượng.
- Thời gian dùng màn hình có thể liên quan đến căng thẳng hoặc gián tiếp ảnh hưởng tới giấc ngủ.
- Mức độ căng thẳng ảnh hưởng trực tiếp đến tâm trạng.
- Thời gian dùng TikTok Có thể chỉ ảnh hưởng gián tiếp đến mood thông qua giấc ngủ hoặc stress.
- Số lượng nền tảng mạng xã hội dùng không ảnh hưởng nhiều đến mood nếu không gắn với thời lượng sử dụng.



Mô hình RandomForestRegressor (Nguyễn Ngọc Hải)

Kết quả sau khi predict



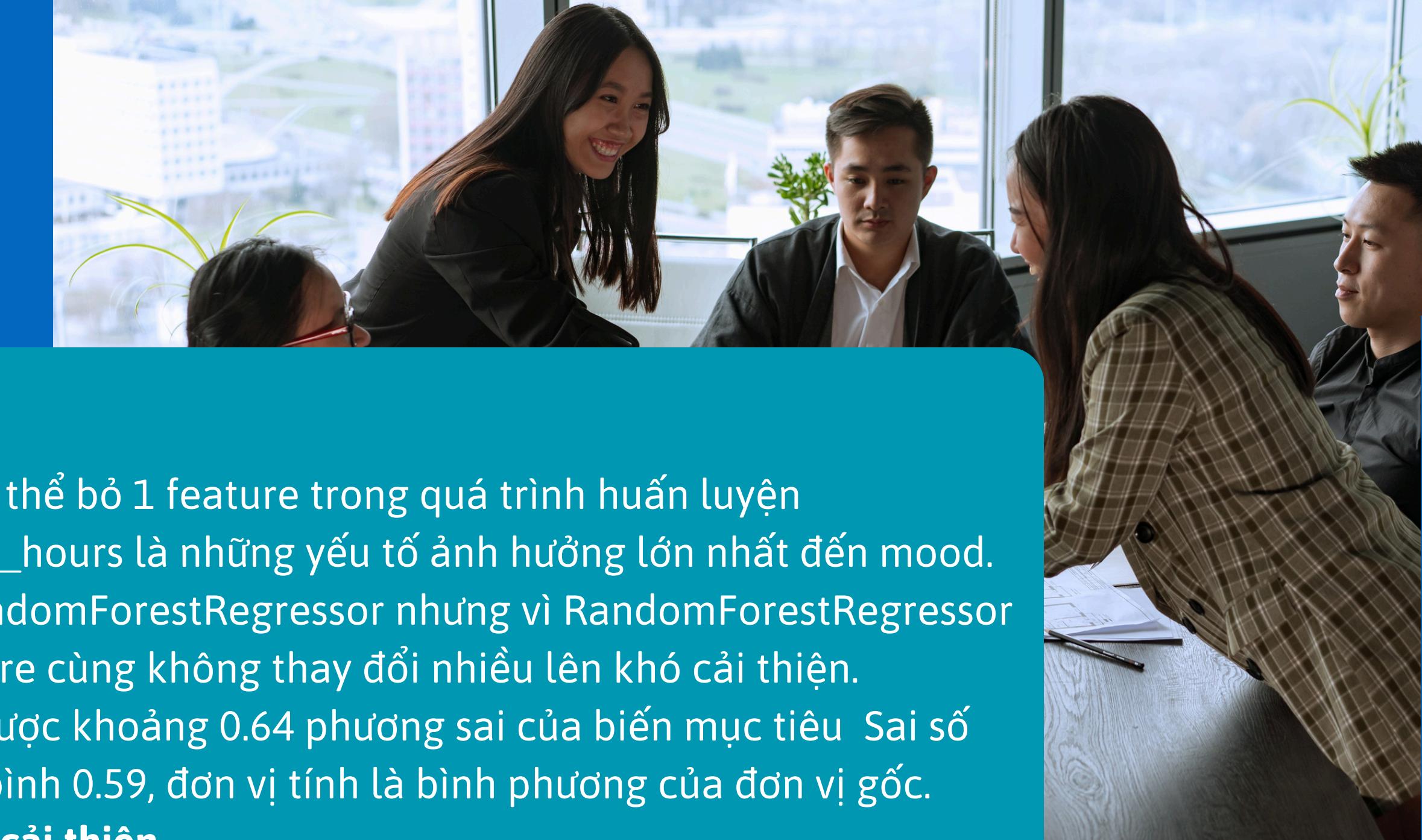
Đánh giá model

```
mseRFR = mean_squared_error(y_test, y_pred)
r2RFR = r2_score(y_test, y_pred)
print(f'MSE: {mseRFR:.2f}')
print(f'R2 Score: {r2RFR:.2f}')
```

MSE: 0.59
R2 Score: 0.64

Sau khi xây kiểm nghiệm với 200 mẫu trong tệp dữ liệu :

- Cả hai đường (thực tế và dự đoán) đều tập trung quanh mức mood từ 7 đến 10, cho thấy mô hình đã học được phần lớn xu hướng.
- Tuy nhiên, vẫn có nhiều điểm bị lệch xa → thể hiện những dự đoán chưa tốt ở một số mẫu, đặc biệt là các điểm có giá trị thực tế thấp (4–6), mô hình thường dự đoán cao hơn.
- Đường dự đoán thường ít rơi xuống các giá trị thấp hơn 6 → điều này cho thấy mô hình có thể thiên lèch về dự đoán các mood_score cao.



📌 Kết luận

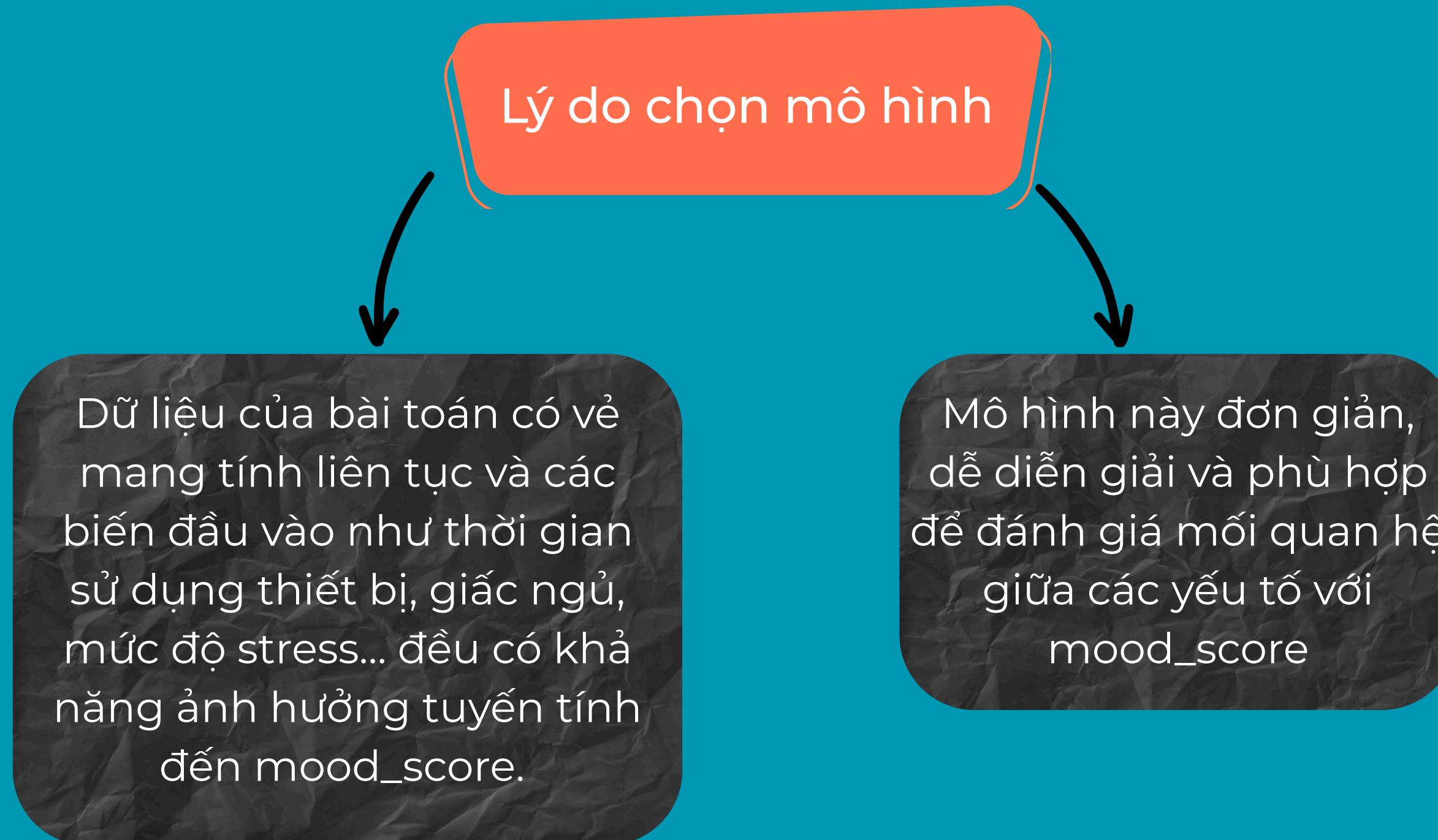
- Xuất hiện outliers có thể bỏ 1 feature trong quá trình huấn luyện
- Stress level và sleep_hours là những yếu tố ảnh hưởng lớn nhất đến mood.
- Lựa chọn model RandomForestRegressor nhưng vì RandomForestRegressor nếu tinh chỉnh feature cùng không thay đổi nhiều lên khó cải thiện.
- Mô hình giải thích được khoảng 0.64 phương sai của biến mục tiêu Sai số bình phương trung bình 0.59, đơn vị tính là bình phương của đơn vị gốc.

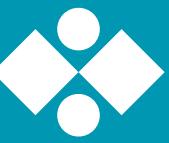
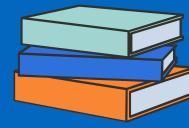
=> **Mô hình tạm ổn, cần cải thiện**



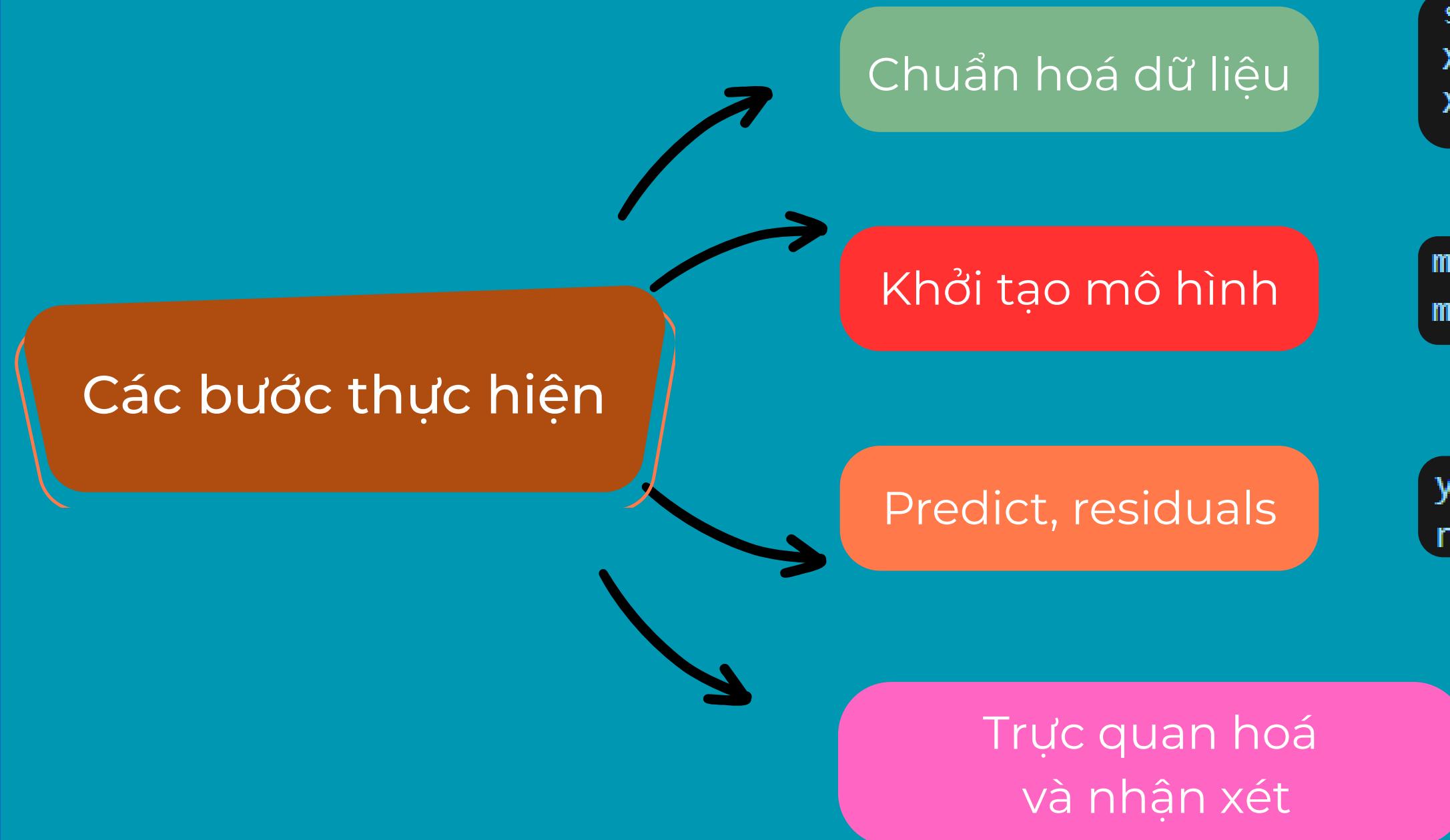


Mô hình Linear Regression(Nguyễn Thành Công)





Mô hình Linear Regression(Nguyễn Thành Công)



```
scaler = StandardScaler()  
X_train_scaled = scaler.fit_transform(X_train)  
X_test_scaled = scaler.transform(X_test)
```

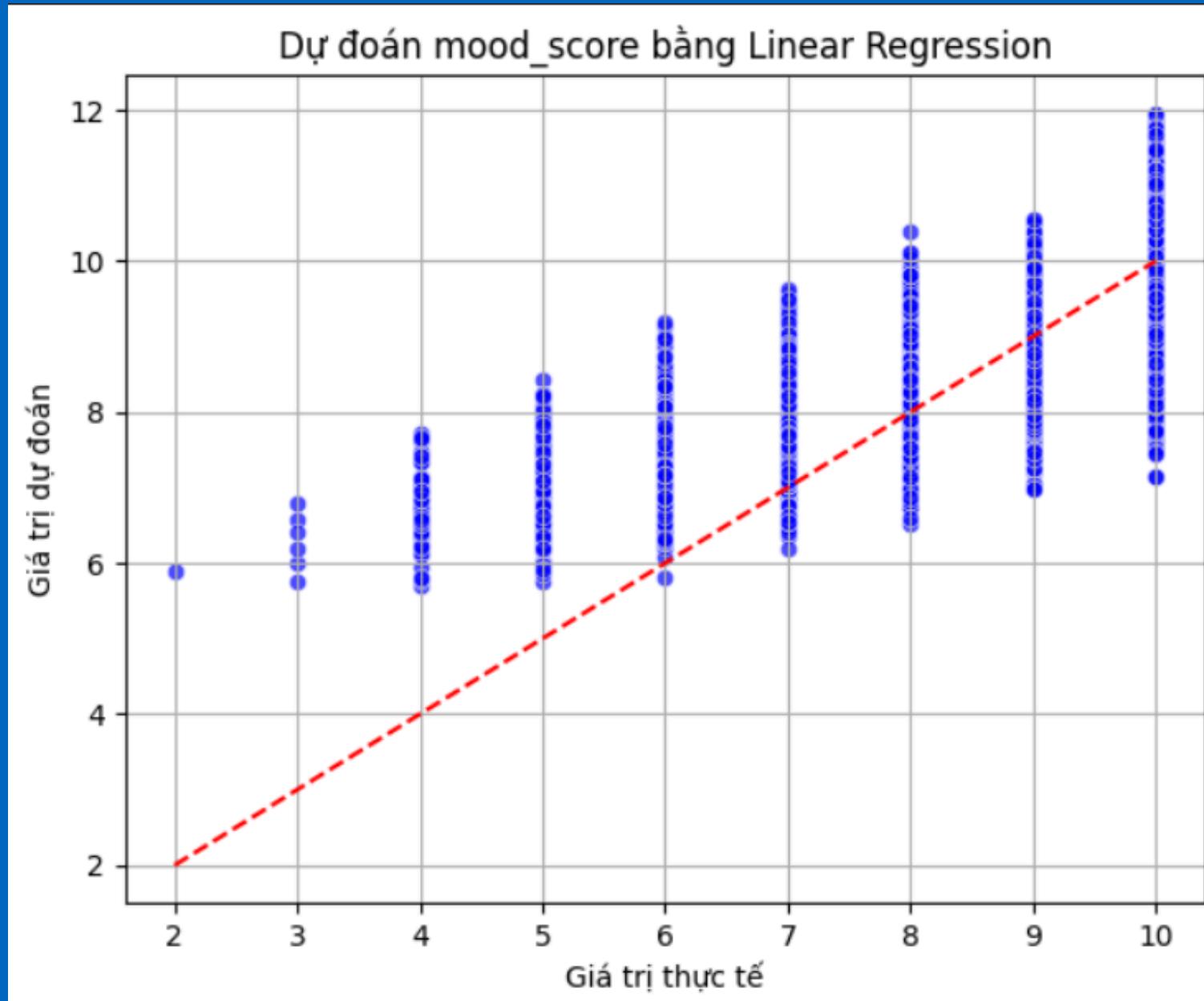
```
model = LinearRegression()  
model.fit(X_train_scaled, y_train)
```

```
y_pred = model.predict(X_test_scaled)  
residuals = y_test - y_pred
```



Mô hình Linear Regression(Nguyễn Thành Công)

Trực quan hóa
và nhận xét



Các điểm xanh (dự đoán) có xu hướng nằm trên đường chéo (dự đoán = thực tế), tuy nhiên:

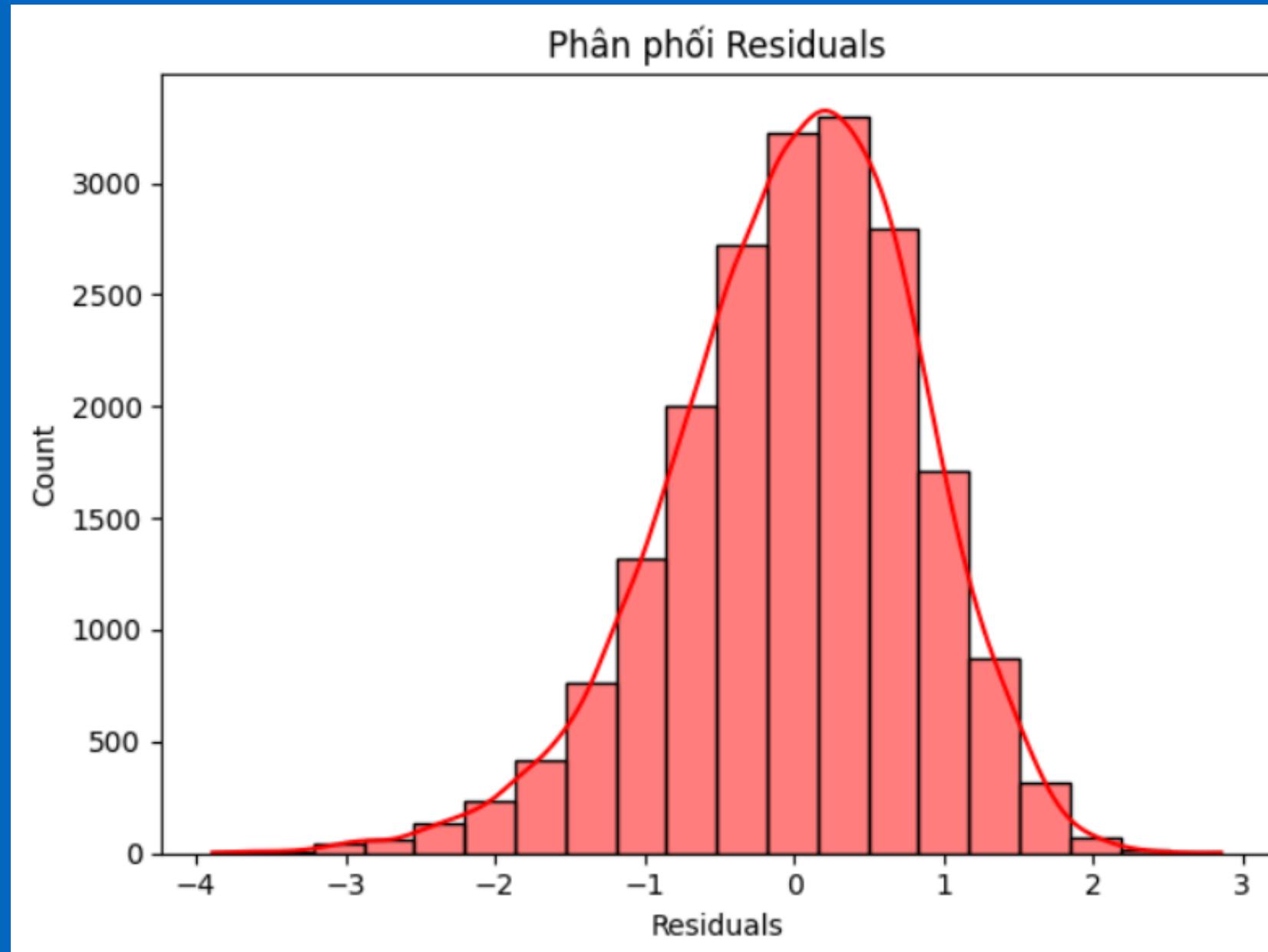
- Mô hình dự đoán cao hơn thực tế với các điểm thấp
- Và có xu hướng dự đoán thấp hơn thực tế với điểm cao

-> Đây là dấu hiệu phổ biến của mô hình tuyến tính khi mô tả một quan hệ có tính phi tuyến tính.



Mô hình Linear Regression(Nguyễn Thành Công)

Trực quan hóa
và nhận xét



- Biểu đồ residual có dạng chuông, gần giống với phân phối chuẩn.
- Sai số tập trung chủ yếu quanh 0, nghĩa là phần lớn dự đoán là khá chính xác.
- Không có dấu hiệu của sai số lệch hoặc độ bất thường
 - Thỏa mãn một giả định quan trọng của hồi quy tuyến tính: phân phối chuẩn của sai số.



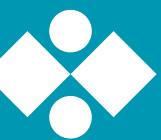
Mô hình Linear Regression(Nguyễn Thành Công)

Kết Luận

- Mean Squared Error (MSE): 0.7004
 - Trung bình bình phương sai số là ~0.7. Giá trị này không quá cao, cho thấy mô hình không quá kém, nhưng vẫn còn độ lệch giữa giá trị dự đoán và thực tế.
- Mean Absolute Error (MAE): 0.6597
 - Trung bình sai số tuyệt đối ~0.66 đơn vị. Với thang đo mood_score là từ 1 đến 10, sai số khoảng hơn nửa điểm là tạm chấp nhận.
- R² Score: 0.5680
 - Mô hình giải thích được khoảng 56.8% phuơng sai của biến mục tiêu. Đây là một mức khá ổn đối với một mô hình tuyến tính trong bài toán hành vi tâm lý, tuy nhiên còn tiềm năng cải thiện bằng các mô hình phi tuyến hoặc kỹ thuật chọn đặc trưng.

Bonus
Sử dụng Cross_val_score

```
R2 scores: [0.56962177 0.57908308 0.57111866 0.57222811 0.5625768 ]  
R2 Mean: 0.5709  
MSE scores: [0.70726601 0.68658155 0.70644819 0.70638084 0.69936417]  
MSE Mean: 0.7012
```



Mô hình Polynomial Regression(Nguyễn Trọng Vỹ)

```
for n in range(2,10):
    poly = PolynomialFeatures(degree=n, include_bias=False)
    X_train_poly = poly.fit_transform(X_train)
    X_test_poly = poly.transform(X_test)

    # Huấn luyện mô hình Polynomial Regression
    model = LinearRegression()
    model.fit(X_train_poly, y_train)

    # Đánh giá mô hình
    print("Đa Thức Bậc:", n)
    y_pred = model.predict(X_test_poly)
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_test, y_pred)
    print("\tMSE:", mse)
    print("\tRMSE:", rmse)
    print("\tR2 Score:", r2)
```

```
Đa Thức Bậc: 2
MSE: 0.5531550826358134
RMSE: 0.7437439630920128
R2 Score: 0.6588118113245954
Đa Thức Bậc: 3
MSE: 0.5494519934144072
RMSE: 0.741250290667334
R2 Score: 0.6610958910404222
Đa Thức Bậc: 4
MSE: 0.5439802839179644
RMSE: 0.7375501907788814
R2 Score: 0.6644708625640563
Đa Thức Bậc: 5
MSE: 0.5445510524284908
RMSE: 0.7379370247036604
R2 Score: 0.6641188103451171
```

Lí do chọn Mô hình Polynomial Regression:

- Cân bằng giữa độ phức tạp cần thiết để học quan hệ phi tuyến trong dữ liệu, và sự đơn giản, dễ triển khai, dễ giải thích.

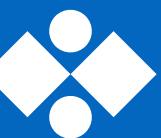
- Với tập dữ liệu có 100.000 dòng và 5 biến đầu vào, mô hình này đủ khả năng học được các mối tương tác phi tuyến.

Tạo vòng lặp từ 2 đến 10 để thử với các đa thức bậc cao để tìm ra đa thức tốt nhất cho mô hình.

Trong vòng lặp thực hiện fit, transform tập train, và transform tập test. Tiếp đến khởi tạo và huấn luyện mô hình với tập train đã được transform. Cuối cùng đánh giá mô hình dựa trên tập test đã được transform và in các thông số ra màn hình.

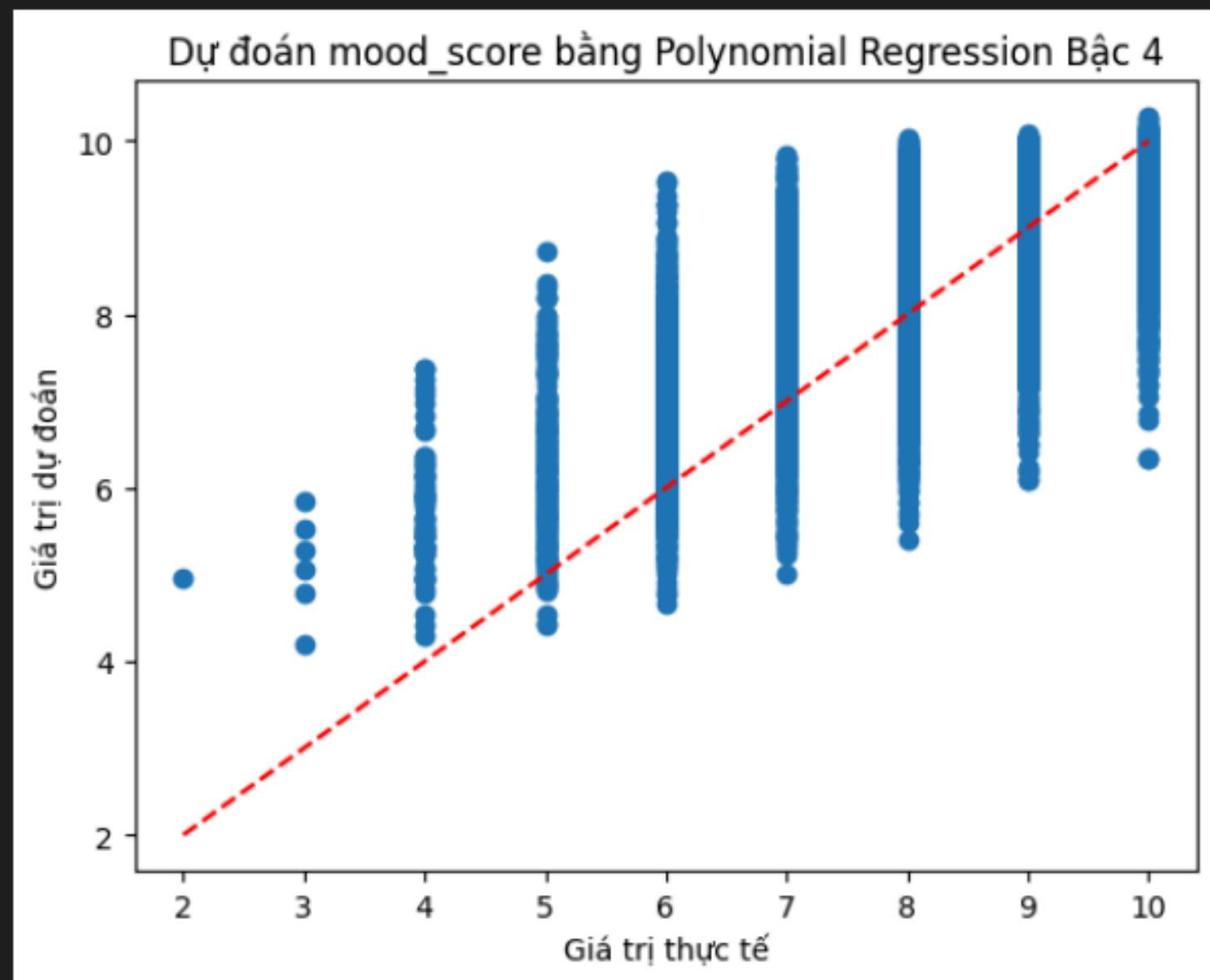
Dựa vào các thông số được in ra, chọn đa thức bậc 4 là hiệu quả nhất với tập dữ liệu này.

Chạy lại khối code trong vòng lặp với degree = 4 để có mô hình cuối cùng.



Mô hình Polynomial Regression(Nguyễn Trọng Vỹ)

```
plt.scatter(y_test, y_pred)
plt.xlabel("Giá trị thực tế")
plt.ylabel("Giá trị dự đoán")
plt.title("Dự đoán mood_score bằng Polynomial Regression Bậc 4")
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--')
plt.show()
```



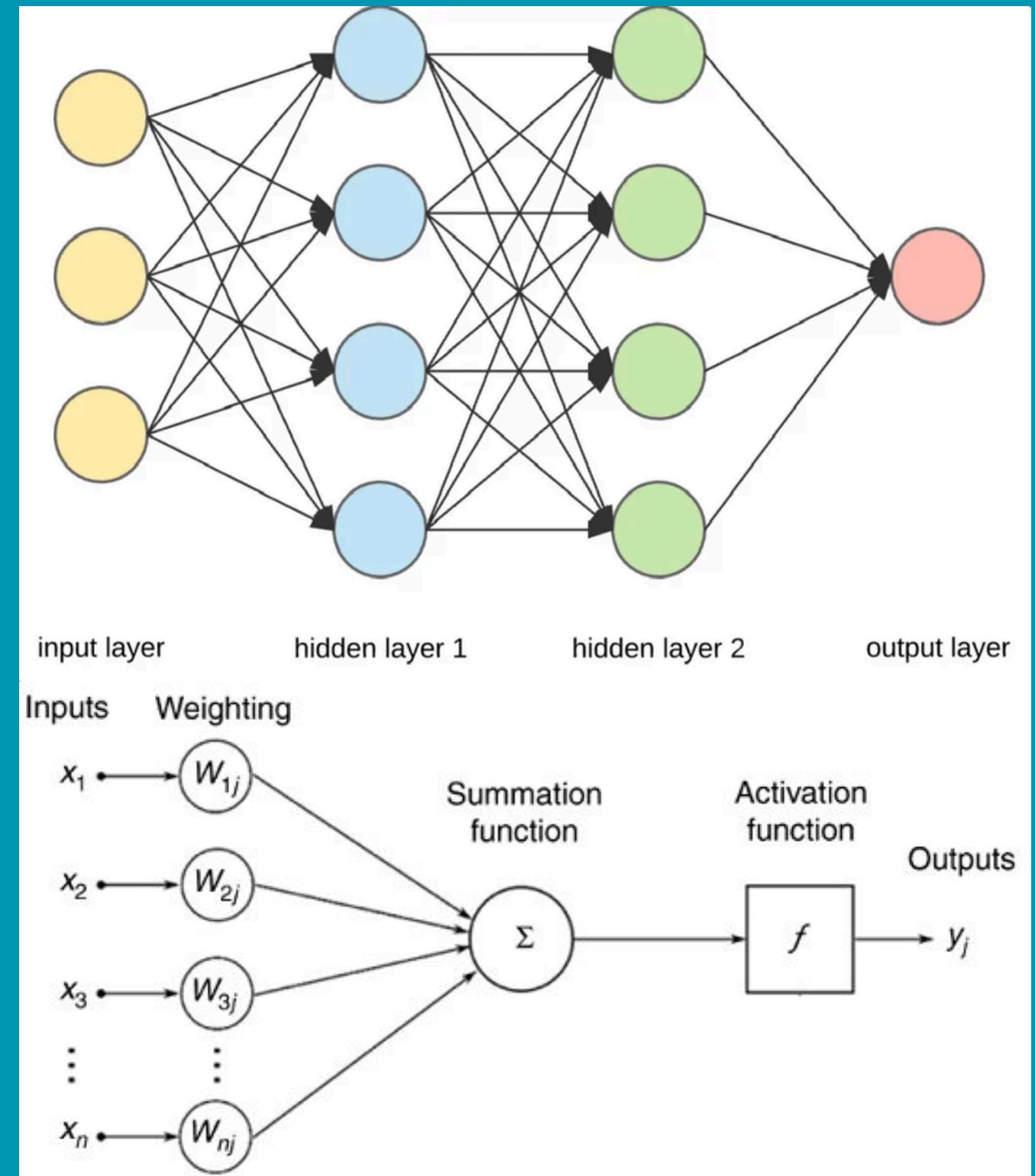
In ra biểu đồ scatter giữa tập test và tập dự đoán để quan sát mô hình.

Nhận thấy mô hình dự đoán mood_score thường cao hơn ở mức mood_score thấp(1-6), và thấp hơn ở mức mood_score cao(9-10).



Mô hình Neural Network Regression(Nguyễn Hoàng Nguyên)

- Là mô hình học máy mô phỏng hoạt động của não người, dùng để dự đoán, phân loại, hoặc nhận dạng mẫu từ dữ liệu.
- Mạng Neural bao gồm: tầng đầu vào, 1 hoặc nhiều tầng ẩn và tầng đầu ra.
- Mỗi tầng ẩn chứa nhiều neuron, dùng để thực hiện phép tính và neuron ở mỗi tầng được kết nối với tất cả neuron ở tầng tiếp theo
- Các trọng số (weights) và bias sẽ được điều chỉnh khi huấn luyện để giảm sai số.





Mô hình Neural Network Regression(Nguyễn Hoàng Nguyên)

Lý do lựa chọn:

- Tập dữ liệu lớn khoảng 100.000 dữ liệu
- Dữ liệu phi tuyến, khá phức tạp

Nhận dữ liệu đầu vào: Các đặc trưng (feature) được đưa vào mạng.

Xử lý qua các tầng ẩn: Dữ liệu được tính toán qua nhiều tầng nơ-ron. Mỗi nơ-ron nhận đầu vào với trọng số, cộng bias rồi đưa qua hàm kích hoạt.

Trả kết quả ở tầng đầu ra: Mạng tạo ra dự đoán cuối cùng là điểm mood_score.

```
# Chuẩn hóa dữ liệu
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Huấn luyện mô hình MLPRegressor
model = MLPRegressor(hidden_layer_sizes=(128, 64, 32),
                      activation='relu',
                      solver='adam', max_iter=2000,
                      early_stopping=True, random_state=42)
model.fit(X_train_scaled, y_train)
```

```
# Đánh giá mô hình
y_pred = model.predict(X_test_scaled)
mseNN = mean_squared_error(y_test, y_pred)
r2NN = r2_score(y_test, y_pred)
print(f'MSE: {mseNN:.4f}')
print(f'R^2: {r2NN:.4f}')
```

✓ 0.0s

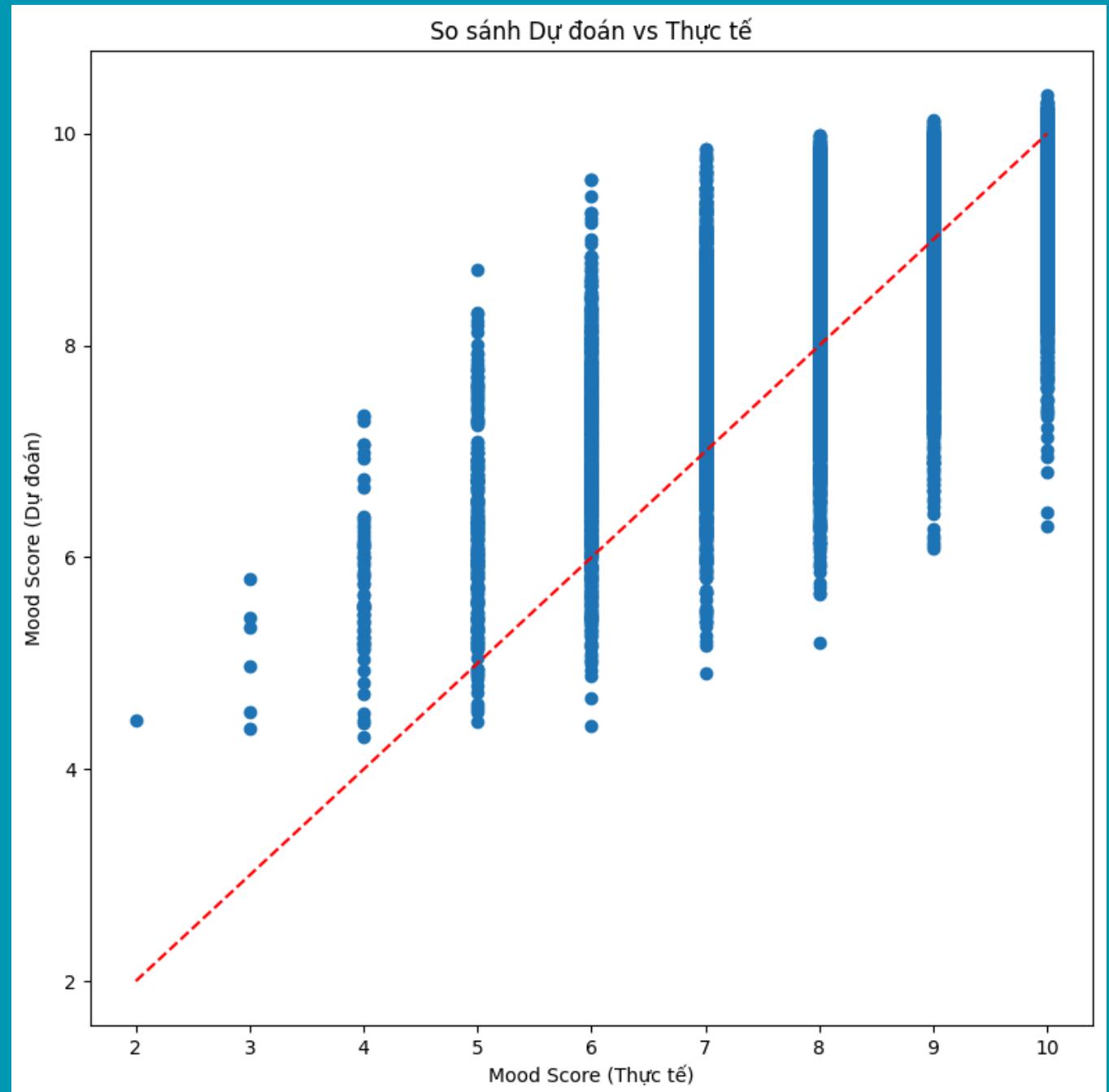
MSE: 0.5489

R²: 0.6615



Mô hình Neural Network Regression(Nguyễn Hoàng Nguyên)

- Mô hình Neural Network huấn luyện được cho độ chính xác không cao (66.15%).
- Dự đoán thiếu chính xác các mức điểm từ 0 đến 5 điểm, có thể do mức điểm thực tế tập trung rất nhiều ở mức 10 điểm khiến xảy ra sai lệch
- Tuy nhiên mô hình vẫn có xu hướng tuyến tính, cho thấy đã có nắm được xu hướng chung.
- Có thể cải thiện thêm bằng cách thêm đặc trưng (feature) hoặc loại bỏ các giá trị outlier.





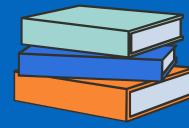
GradientBoostingRegressor(Vũ Minh Đức)



Lý do chọn GradientBoostingRegressor

- Đây là mô hình hồi quy mạnh, phù hợp để dự đoán mood_score – một giá trị liên tục.
- Mô hình hóa tốt các mối quan hệ phức tạp và phi tuyến giữa các yếu tố (ví dụ: sleep_hours, stress_level, screen_time_hours...) với mood.
- Mô hình còn cung cấp feature importance, giúp giải thích rõ yếu tố nào ảnh hưởng mạnh nhất.





GradientBoostingRegressor(Vũ Minh Đức)



Tiền xử lí dữ liệu

```
from sklearn.preprocessing import StandardScaler  
  
# Khởi tạo bộ chuẩn hóa  
scaler = StandardScaler()  
  
# Fit scaler trên dữ liệu train và transform  
X_train_scaled = scaler.fit_transform(X_train)  
  
# Transform dữ liệu test (dùng scaler đã fit)  
X_test_scaled = scaler.transform(X_test)  
  
# Kiểm tra kích thước  
print("✓ X_train_scaled shape:", X_train_scaled.shape)  
print("✓ X_test_scaled shape:", X_test_scaled.shape)
```

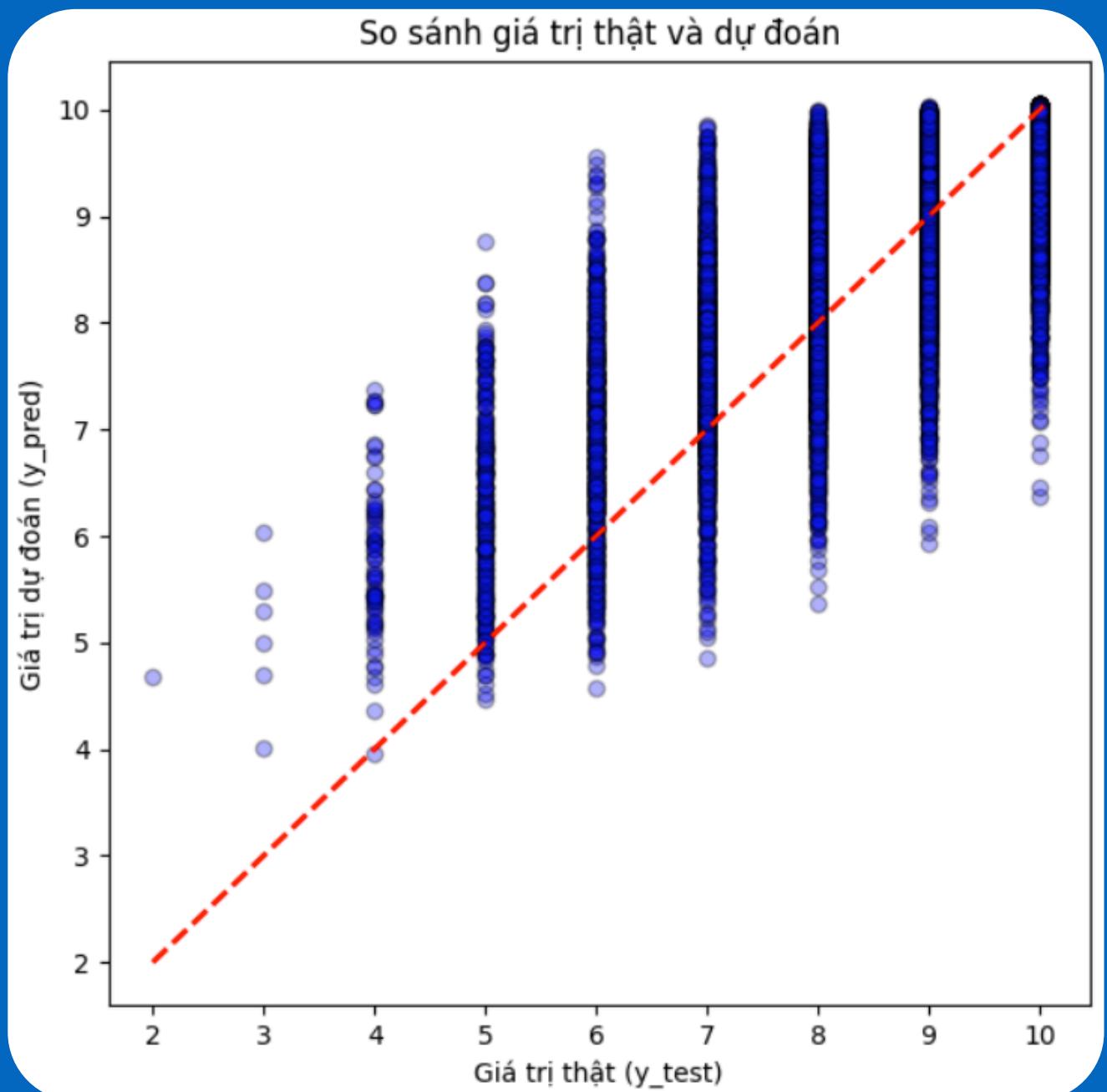
Huấn luyện model

```
# BƯỚC 4: Huấn luyện mô hình GradientBoostingRegressor  
from sklearn.ensemble import GradientBoostingRegressor  
  
# Khởi tạo mô hình  
gbr = GradientBoostingRegressor(  
    n_estimators=200,      # số lượng cây  
    learning_rate=0.1,    # tốc độ học  
    max_depth=3,         # độ sâu của mỗi cây  
    random_state=42  
)  
  
# Huấn luyện mô hình  
gbr.fit(X_train_scaled, y_train)
```



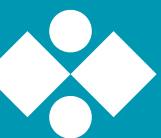
GradientBoostingRegressor(Vũ Minh Đức)

So sánh kết quả sau khi predict



Biểu đồ scatter giữa giá trị thật và dự đoán cho thấy:

- Xu hướng dự đoán của mô hình GradientBoostingRegressor phù hợp với xu hướng dữ liệu thật.
- Đa số điểm nằm gần đường $y=x$ (đường đỏ), chứng tỏ mô hình dự đoán chính xác khá cao.
- Một số trường hợp giá trị thật thấp (mood_score 2–4) bị dự đoán hơi cao, thể hiện sai số nhỏ ở vùng thấp, nhưng không ảnh hưởng lớn đến tổng thể.
- Tổng quan, mô hình hoạt động tốt và có khả năng dự đoán đáng tin cậy.



GradientBoostingRegressor(Vũ Minh Đức)

Đánh giá model

```
# BƯỚC 5: Đánh giá mô hình
# Dự đoán
y_pred = gbr.predict(x_test_scaled)

mseGBR = mean_squared_error(y_test, y_pred)
r2GBR = r2_score(y_test, y_pred)

print(f"➤ Mean Squared Error (MSE): {mseGBR:.2f}")
print(f"➤ R² Score: {r2GBR:.2f}")
```

```
Mean Squared Error (MSE): 0.55
R² Score: 0.66
```

Phân tích kết quả để ghi vào báo cáo:

MSE = 0.55

Sai số trung bình bình phương giữa giá trị thật và giá trị dự đoán khá thấp.

Mô hình dự đoán tốt, sai số nhỏ.

R² = 0.66

Mô hình giải thích được 66% phương sai của dữ liệu.

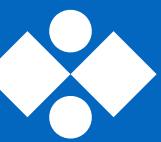
Đây là một kết quả khá tốt cho mô hình Gradient Boosting ở lần huấn luyện đầu tiên.

Đánh giá mô hình:

MSE = 0.55 (thấp, dự đoán chính xác tương đối tốt)

R² = 0.66 (mô hình giải thích 66% sự biến thiên của dữ liệu)

Mô hình GradientBoostingRegressor đã đạt kết quả khả quan trên tập dữ liệu này.



ĐÁNH GIÁ CHUNG GIỮA CÁC MÔ HÌNH

Tổng hợp kết quả từ các model

```
==== R2 Scores cho từng mô hình ===
Linear_Regression: 0.5680
GradientBoostingRegressor: 0.6626
RandomForestRegressor: 0.6383
neural_network: 0.6615
Polynomial_Regression: 0.6645
```

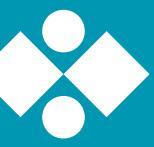
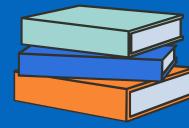
```
==== MSE cho từng mô hình ===
Linear_Regression: 0.7004
GradientBoostingRegressor: 0.5469
RandomForestRegressor: 0.5864
neural_network: 0.5489
Polynomial_Regression: 0.5440
```

📈 R² Scores (độ chính xác):

- Polynomial Regression: 0.6645 — cao nhất, cho thấy mô hình này mô phỏng tốt quan hệ phi tuyến giữa dữ liệu đầu vào và đầu ra.
- GradientBoostingRegressor: 0.6626 — cũng rất tốt, chỉ kém một chút.
- Neural Network: 0.6615 — khá ổn, nhưng không vượt trội.
- RandomForestRegressor: 0.6383 — thấp hơn một chút.
- Linear Regression: 0.5680 — thấp nhất, do không phù hợp với quan hệ phi tuyến.

📉 MSE (Mean Squared Error – độ sai số):

- Polynomial Regression: 0.5440 — nhỏ nhất, chứng tỏ dự đoán sát thực tế hơn cả.
- Neural Network: 0.5489
- GradientBoosting: 0.5469
- Random Forest: 0.5864
- Linear Regression: 0.7004 — sai số cao nhất.



ĐÁNH GIÁ CHUNG GIỮA CÁC MÔ HÌNH

Chọn model

- => Mô hình được đề xuất sử dụng: Polynomial Regression
- Có R^2 cao nhất và MSE thấp nhất trong tất cả các mô hình đã thử.
- Hiệu quả vượt trội hơn Linear Regression nhờ khả năng mô hình hóa quan hệ phi tuyến.
- Dễ cài đặt, dễ huấn luyện.
- Kiểm soát được mức độ phức tạp qua tham số degree.

Hướng phát triển

- Tối ưu bậc đa thức (degree): thử các giá trị từ 2 → 5 để tìm điểm cân bằng giữa độ chính xác và tránh quá khớp (overfitting).
- Chuẩn hóa dữ liệu đầu vào (nếu chưa làm): dùng StandardScaler giúp Polynomial Features ổn định hơn.
- Cross-validation: Áp dụng k-fold cross-validation để kiểm tra độ ổn định và khả năng tổng quát của mô hình.
- Loại bỏ outlier khi huấn luyện
- Tiếp tục tìm hiểu các model khác để cải thiện bài toán



Thank you