

# BÀI TUẦN 04: EVALUATING REGRESSION MODELS PERFORMANCE

## 1. Thông tin sinh viên

DƯƠNG MINH LƯỢNG-18521071

## 2. Source

### 1. Linear Regression

```
1. import matplotlib.pyplot as plt
2. import pandas as pd
3.
4. # dataset = pd.read_csv('Position_Salaries.csv')
5. dataset_train = pd.read_csv('Position_SalariesTrain.csv')
6. dataset_test = pd.read_csv('Position_SalariesTest.csv')
7. X = dataset_train.iloc[:, 1:-1].values
8. Y = dataset_train.iloc[:, -1].values
9. X_test = dataset_test.iloc[:, 1:-1].values
10. Y_test = dataset_test.iloc[:, -1].values
11.
12. from sklearn.linear_model import LinearRegression
13. lin_reg = LinearRegression()
14. lin_reg.fit(X, Y)
15. Y_pred = lin_reg.predict(X)
16. Y_pred_test = lin_reg.predict(X_test)
17. plt.scatter(X_test, Y_test, color = "red")
18. plt.scatter(X_test, Y_pred_test, color = "black")
19. plt.plot(X, Y_pred, color = "blue")
20. plt.title("Position Level vs Salary (Linear Regression)")
21. plt.xlabel("Position Level")
22. plt.ylabel("Salary (dollars/year)")
23. plt.show()
24.
25. from sklearn.metrics import mean_squared_error
26. from math import sqrt
27. print("SSE", len(X_test)*mean_squared_error(Y_test, Y_pred_test))
28. print("RMSE", sqrt(mean_squared_error(Y_test, Y_pred_test)))
29. from sklearn.metrics import r2_score
30. r2=r2_score(Y_test, Y_pred_test)
31. print("r2= ",r2)
```

```
32. adjusted_r_squared = 1 - (1-r2)*((len(Y_test)-1)/(len(Y_test)-X_test.shape[1]-1))
33. print("adjusted_r_squared",adjusted_r_squared)
34. import statsmodels.regression.linear_model as sm
35. regressor_OLS = sm.OLS(endog = Y, exog = X).fit()
36. print(regressor_OLS.summary())
```

## 2. Polynomial Regression

```
1. import numpy as np
2. import matplotlib.pyplot as plt
3. import pandas as pd
4.
5. # dataset = pd.read_csv('Position_Salaries.csv')
6. dataset_train = pd.read_csv('Position_SalariesTrain.csv')
7. dataset_test = pd.read_csv('Position_SalariesTest.csv')
8. X = dataset_train.iloc[:, 1:-1].values
9. Y = dataset_train.iloc[:, -1].values
10. X_test = dataset_test.iloc[:, 1:-1].values
11. Y_test = dataset_test.iloc[:, -1].values
12.
13. from sklearn.preprocessing import PolynomialFeatures
14. poly_transform = PolynomialFeatures(degree=4)
15. X_poly = poly_transform.fit_transform(X)
16. X_poly_test = poly_transform.fit_transform(X_test)
17.
18. from sklearn.linear_model import LinearRegression
19. poly_lin_reg = LinearRegression()
20. poly_lin_reg.fit(X_poly, Y)
21.
22. Y_poly_pred = poly_lin_reg.predict(X_poly)
23. Y_poly_pred_test = poly_lin_reg.predict(X_poly_test)
24. plt.scatter(X_test, Y_test ,color = "red")
25. plt.plot(X, Y_poly_pred, color = "blue")
26. plt.scatter(X_test, Y_poly_pred_test ,color = "black")
27. plt.title("Position Level vs Salary(Polynomial Regression)")
28. plt.xlabel("Position Level")
29. plt.ylabel("Salary (dollars/year)")
30. plt.show()
31.
32. X_grid = np.arange(0, max(X)+1, 0.1)
33. X_grid = X_grid.reshape((len(X_grid), 1))
34. plt.scatter(X_test, Y_test, color = "red")
35. plt.plot(X_grid, poly_lin_reg.predict(poly_transform.fit_transform(X_grid)),
           color = 'blue')
36. plt.scatter(X_test, Y_poly_pred_test, color = "black")
```

```

37. plt.title("Position Level vs Salary(Polynomial Regression)")
38. plt.xlabel("Position Level")
39. plt.ylabel("Salary (dollars/year)")
40. plt.show()
41.
42.
43. # from sklearn.metrics import mean_absolute_error
44. # print("MAE", mean_absolute_error(Y_test, Y_poly_pred_test))
45. from sklearn.metrics import mean_squared_error
46. from math import sqrt
47. print("SSE", len(X_test)*mean_squared_error(Y_test, Y_poly_pred_test))
48. print("RMSE", sqrt(mean_squared_error(Y_test, Y_poly_pred_test)))
49. from sklearn.metrics import r2_score
50. r2=r2_score(Y_test, Y_poly_pred_test)
51. print("r2=", r2)
52. adjusted_r_squared = 1 - (1-r2)*((len(Y_test)-1)/(len(Y_test)-X_test.shape[1]-1))
53. print("adjusted_r_squared= ", adjusted_r_squared)
54. import statsmodels.regression.linear_model as sm
55. regressor_OLS = sm.OLS(endog = Y, exog = X).fit()
56. print(regressor_OLS.summary())

```

### 3. Support Vector Regression (SVR)

```

1. import numpy as np
2. import matplotlib.pyplot as plt
3. import pandas as pd
4.
5. # dataset = pd.read_csv('Position_Salaries.csv')
6. dataset_train = pd.read_csv('Position_SalariesTrain.csv')
7. dataset_test = pd.read_csv('Position_SalariesTest.csv')
8. X = dataset_train.iloc[:, 1:-1].values
9. Y = dataset_train.iloc[:, -1].values
10. X_test = dataset_test.iloc[:, 1:-1].values
11. Y_test = dataset_test.iloc[:, -1].values
12. Y = Y.reshape(len(Y), 1)
13. Y_test = Y_test.reshape(len(Y_test), 1)
14. from sklearn.preprocessing import StandardScaler
15. sc_X = StandardScaler()
16. sc_y = StandardScaler()
17. X_trans = sc_X.fit_transform(X)
18. Y_trans = sc_y.fit_transform(Y)
19. from sklearn.svm import SVR
20. regressor = SVR(kernel = 'rbf')
21. regressor.fit(X_trans, Y_trans)

```

```

22. def predict(model, X, SC_X, SC_Y):
23.     X_trans = SC_X.transform(X)
24.     Y_trans_pred = model.predict(X_trans)
25.     Y_pred = SC_Y.inverse_transform(Y_trans_pred)
26.     return Y_pred
27. Y_pred_train = predict(regressor, X, sc_X, sc_y)
28. Y_pred_test = predict(regressor, X_test, sc_X, sc_y)
29. plt.scatter(X_test, Y_test, color = 'red')
30. plt.scatter(X_test, Y_pred_test, color = 'black')
31. plt.plot(X, Y_pred_train, color = 'blue')
32. plt.title("Truth or Bluff (SVR)")
33. plt.xlabel('Position level')
34. plt.ylabel('Salary')
35. plt.show()
36.
37. X_grid = np.arange(0, 11, 0.1)
38. X_grid= X_grid.reshape((len(X_grid), 1))
39. Y_pred_grid = predict(regressor, X_grid, sc_X, sc_y)
40. plt.scatter(X_test, Y_test, color = 'red')
41. plt.scatter(X_test, Y_pred_test, color = 'black')
42. plt.plot(X_grid, Y_pred_grid, color = 'blue')
43. plt.title("Truth or Bluff (SVR)")
44. plt.xlabel('Position level')
45. plt.ylabel('Salary')
46. plt.show()
47.
48. from sklearn.metrics import mean_squared_error
49. from math import sqrt
50. print("SSE",len(X_test)*mean_squared_error(Y_test, Y_pred_test))
51. print("RMSE", sqrt(mean_squared_error(Y_test, Y_pred_test)))
52. from sklearn.metrics import r2_score
53. r2=r2_score(Y_test, Y_pred_test)
54. print("r2=",r2)
55. adjusted_r_squared = 1 - (1-r2)*((len(Y_test)-1)/(len(Y_test)-X_test.shape[1]-1))
56. print("adjusted_r_squared= ",adjusted_r_squared)

```

#### 4. Decision Tree Regression

```

1. import numpy as np
2. import matplotlib.pyplot as plt
3. import pandas as pd
4.
5. # dataset = pd.read_csv('Position_Salaries.csv')
6. dataset_train = pd.read_csv('Position_SalariesTrain.csv')
7. dataset_test = pd.read_csv('Position_SalariesTest.csv')

```

```

8. X = dataset_train.iloc[:, 1:-1].values
9. Y = dataset_train.iloc[:, -1].values
10. X_test = dataset_test.iloc[:, 1:-1].values
11. Y_test = dataset_test.iloc[:, -1].values
12. Y = Y.reshape(len(Y),1)
13. Y_test = Y_test.reshape(len(Y_test),1)
14. from sklearn.tree import DecisionTreeRegressor
15. regressor = DecisionTreeRegressor(random_state = 0)
16. regressor.fit(X, Y)
17. Y_pred_test=regressor.predict(X_test)
18. X_grid = np.arange(min(X), max(X), 0.01)
19. X_grid = X_grid.reshape((len(X_grid), 1))
20. plt.scatter(X_test, Y_test, color = 'red')
21. plt.scatter(X_test, Y_pred_test, color = 'black')
22. plt.plot(X_grid, regressor.predict(X_grid), color = 'blue')
23. plt.title('Truth or Bluff (Decision Tree Regression)')
24. plt.xlabel('Position level')
25. plt.ylabel('Salary')
26. plt.show()
27.
28. from sklearn.metrics import mean_squared_error
29. from math import sqrt
30. print("SSE",len(X_test)*mean_squared_error(Y_test, Y_pred_test))
31. print("RMSE", sqrt(mean_squared_error(Y_test, Y_pred_test)))
32. from sklearn.metrics import r2_score
33. r2=r2_score(Y_test, Y_pred_test)
34. print("r2=",r2)
35. adjusted_r_squared = 1 - (1-r2)*((len(Y_test)-1)/(len(Y_test)-
    X_test.shape[1]-1))
36. print("adjusted_r_squared= ",adjusted_r_squared)

```

## 5. Random Forest Regression

```

1. import numpy as np
2. import matplotlib.pyplot as plt
3. import pandas as pd
4.
5. # dataset = pd.read_csv('Position_Salaries.csv')

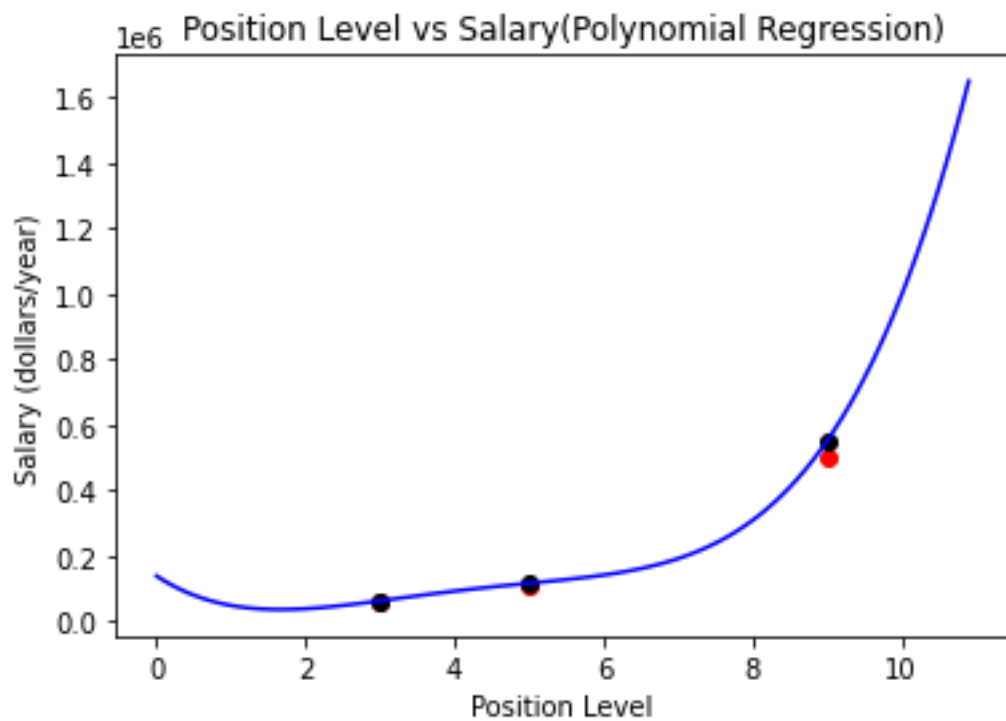
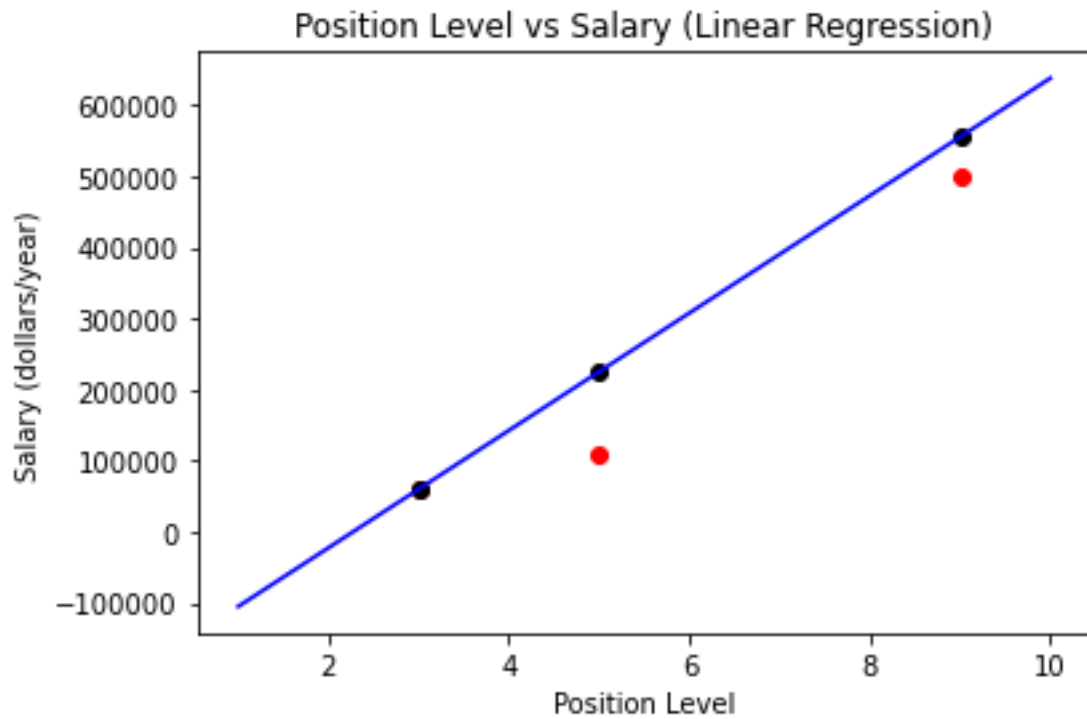
```

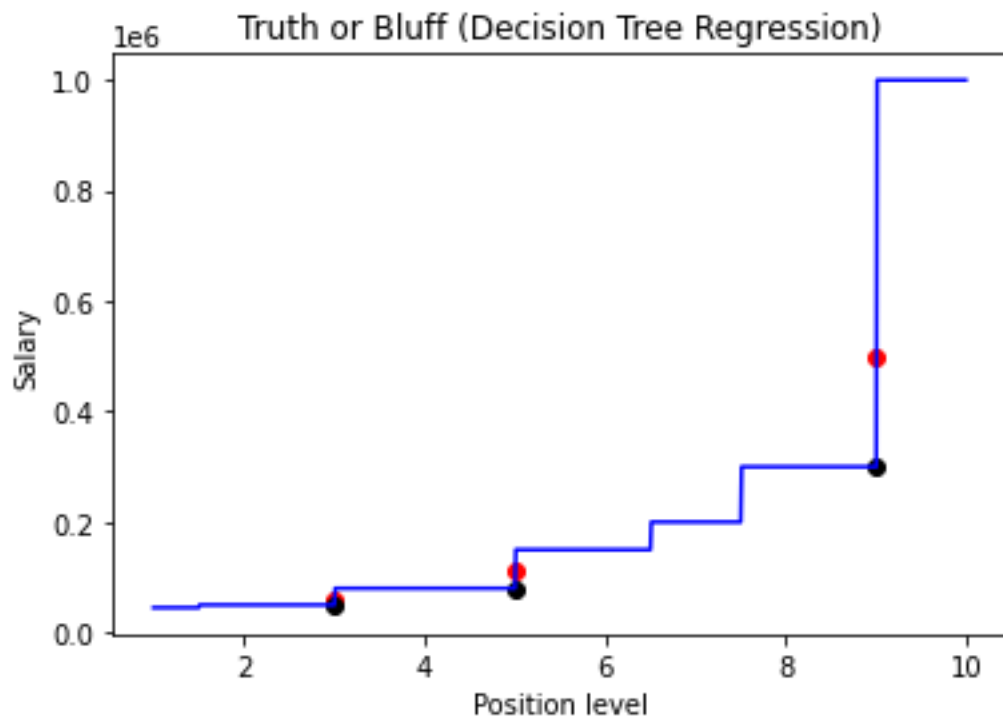
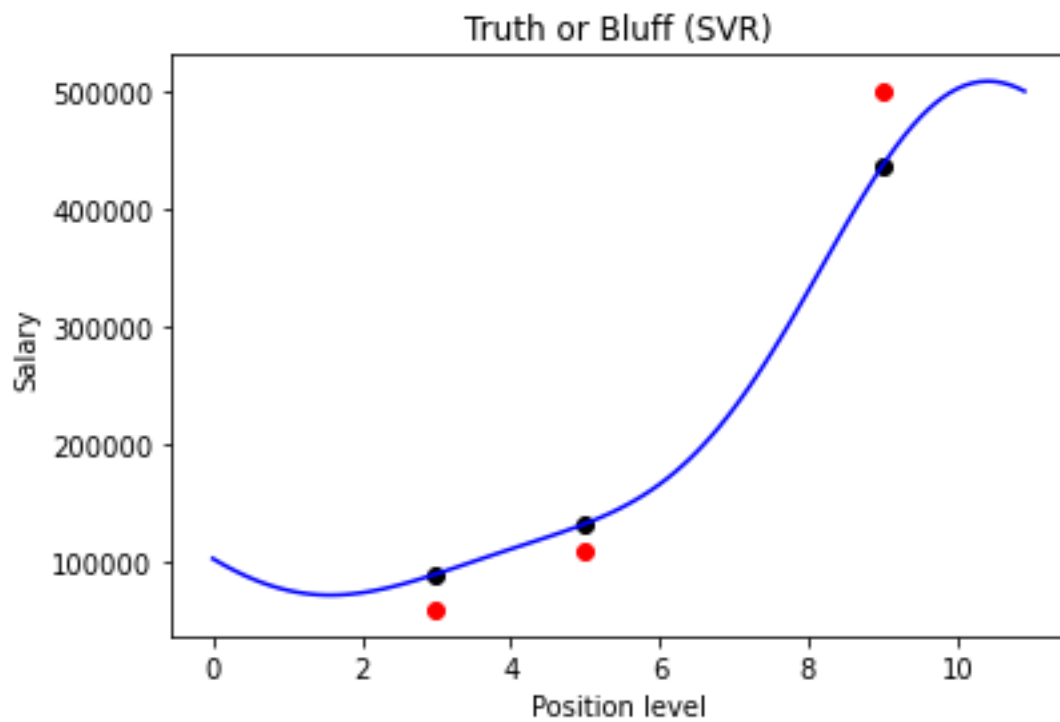
```

6. dataset_train = pd.read_csv('Position_SalariesTrain.csv')
7. dataset_test = pd.read_csv('Position_SalariesTest.csv')
8. X = dataset_train.iloc[:, 1:-1].values
9. Y = dataset_train.iloc[:, -1].values
10. X_test = dataset_test.iloc[:, 1:-1].values
11. Y_test = dataset_test.iloc[:, -1].values
12. Y = Y.reshape(len(Y),1)
13. Y_test = Y_test.reshape(len(Y_test),1)
14.
15. from sklearn.ensemble import RandomForestRegressor
16. regressor = RandomForestRegressor(n_estimators = 7 , random_state =
    0 )
17. regressor.fit(X, Y)
18. Y_pred_test=regressor.predict(X_test)
19. X_grid = np.arange(min(X), max(X), 0.1)
20. X_grid = X_grid.reshape((len(X_grid), 1))
21. plt.scatter(X_test, Y_test, color = 'red')
22. plt.scatter(X_test, Y_pred_test, color = 'black')
23. plt.plot(X_grid, regressor.predict(X_grid), color = 'blue')
24. plt.title('Truth or Bluff (Random Forest Regression)')
25. plt.xlabel('Position level')
26. plt.ylabel('Salary')
27. plt.show()
28.
29. from sklearn.metrics import mean_squared_error
30. from math import sqrt
31. print("SSE",len(X_test)*mean_squared_error(Y_test, Y_pred_test))
32. print("RMSE", sqrt(mean_squared_error(Y_test, Y_pred_test)))
33. from sklearn.metrics import r2_score
34. r2=r2_score(Y_test, Y_pred_test)
35. print("r2=",r2)
36. adjusted_r_squared = 1 - (1-r2)*((len(Y_test)-1)/(len(Y_test)-
    X_test.shape[1]-1))
37. print("adjusted_r_squared= ",adjusted_r_squared)

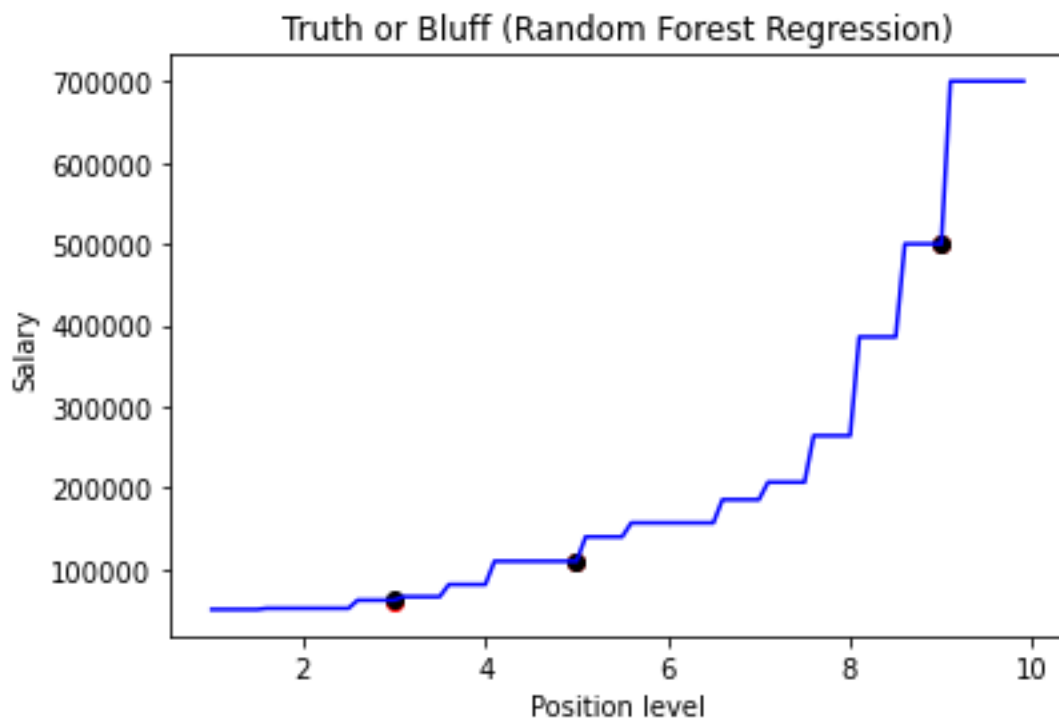
```

### 3. Kết quả









	Squared Sum (SSE)	RMSE	R^2	R_adjusted^2
Linear Regression	16378430468.137295	73888.27256549196	0.8588877328994489	0.7177754657988977
Polynomial Regression	2839073339.221048	30762.928660001624	0.9755392877149249	0.9510785754298499
Support Vector Regression (SVR)	5277090661.4986925	41940.79422828762	0.9545339690278688	0.9090679380557376
Decision Tree Regression	41000000000.0	116904.5194450012	0.6467547386559449	0.2935094773118898

Random Forest Regression	<b>4591836.734693887</b>	<b>1237.179148263485</b>	<b>0.999960437937380</b>	<b>0.9999208758747611</b>
--------------------------	--------------------------	--------------------------	--------------------------	---------------------------

Nhận xét:

- Với Hàm lỗi Squared Sum (SSE) thì phương pháp Random Forest Regression là tốt nhất.
- Với Hàm lỗi Root Mean Squared (RMSE) thì phương pháp Random Forest Regression là tốt nhất.
- Với Hàm đánh giá  $R^2$  thì phương pháp Random Forest Regression là tốt nhất.
- Với Hàm đánh giá  $R_{\text{adjusted}}^2$  thì phương pháp Random Forest Regression là tốt nhất.
- Với tập dữ liệu và test trên thì với phương pháp Random Forest Regression thì cho tất cả các đánh giá độ đo là tốt nhất.
- Với việc `random_state = 0` thì cho ta thấy được phương pháp Random Forest Regression là tốt nhất.
- Với tập dữ liệu và test trên thì với phương pháp Decision Tree Regression thì cho tất cả các đánh giá độ đo là tệ nhất.
- Nếu không có `random_state = 0` thì sẽ cho những kết quả khác nhau sau mỗi lần chạy.

Run 1:

```
SSE 2177551020.408165
RMSE 26941.60982822027
r2= 0.9812387907489245
adjusted_r_squared= 0.962477581497849
```

Run 2:

```
SSE 206632653.0612244
RMSE 8299.25002758732
r2= 0.9982197071821262
adjusted_r_squared= 0.9964394143642523
```

Run 3:

```
SSE 45922959183.67346  
RMSE 123724.10056745002  
r2= 0.6043398117431924  
adjusted_r_squared= 0.2086796234863848
```

Run 4:

```
SSE 13163265306.122444  
RMSE 66240.13211068358  
r2= 0.8865887538243328  
adjusted_r_squared= 0.7731775076486656
```

Run 5:

```
SSE 10002040816.32653  
RMSE 57740.917947692666  
r2= 0.9138250360454349  
adjusted_r_squared= 0.8276500720908697
```

- Với việc chạy ngẫu nhiên 5 lần với việc không dùng `random_state = 0` thì kết quả hoàn toàn khác nhau không tốt nhất với một số lần chạy trên. Với Run 3 thì có lẽ là kết quả tệ nhất với 5 phương pháp.