**BÀI TUẦN 04: EVALUATING REGRESSION MODELS**

**PERFORMANCE**

**1.Thông tin sinh viên**

DƯƠNG MINH LƯỢNG-18521071

# 2.Source

1. **Linear Regression**

|  |
| --- |
| 1. import pandas as pd 2. import numpy as np 3. from sklearn.linear\_model import LinearRegression 4. from sklearn.preprocessing import OneHotEncoder 5. from sklearn.compose import ColumnTransformer 6. # dataset = pd.read\_csv('Position\_Salaries.csv') 7. dataset\_train = pd.read\_csv('city\_day\_AQI\_train.csv') 8. dataset\_test = pd.read\_csv('city\_day\_AQI\_test.csv') 9. del dataset\_train['Date'] 10. del dataset\_test['Date'] 11. dataset\_train.fillna(dataset\_train.groupby('City').transform('mean'),inplace=True) 12. dataset\_test.fillna(dataset\_test.groupby('City').transform('mean'),inplace=True) 13. dataset\_train = dataset\_train.fillna(dataset\_train.mean()) 14. dataset\_test = dataset\_test.fillna(dataset\_test.mean()) 15. X = dataset\_train.iloc[:, :-2].values 16. Y = dataset\_train.iloc[:, -2].values 17. X\_test = dataset\_test.iloc[:,:-2].values 18. Y\_test = dataset\_test.iloc[:, -2].values 19. a=X\_test 20. ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0])], remainder='passthrough') 21. X = np.array(ct.fit\_transform(X)) 22. X\_test = np.array(ct.transform(X\_test)) 23. lin\_reg = LinearRegression() 24. lin\_reg.fit(X, Y) 25. Y\_test\_pred = lin\_reg.predict(X\_test) 26. from sklearn.metrics import mean\_squared\_error 27. from sklearn.metrics import r2\_score 28. from math import sqrt 29. print("SSE",len(X\_test)\*mean\_squared\_error(Y\_test,Y\_test\_pred)) 30. print("RMSE", sqrt(mean\_squared\_error(Y\_test, Y\_test\_pred))) 31. r2 = r2\_score(Y\_test, Y\_test\_pred) 32. print("r2=",r2) 33. adjusted\_r\_squared = 1 - (1-r2)\*((len(Y\_test)-1)/(len(Y\_test)-a.shape[1]-1)) 34. print("adjusted\_r\_squared= ",adjusted\_r\_squared) |

1. **Polynomial Regression**

|  |
| --- |
| 1. import pandas as pd 2. import numpy as np 3. from sklearn.linear\_model import LinearRegression 4. from sklearn.preprocessing import OneHotEncoder 5. from sklearn.compose import ColumnTransformer 6. # dataset = pd.read\_csv('Position\_Salaries.csv') 7. dataset\_train = pd.read\_csv('city\_day\_AQI\_train.csv') 8. dataset\_test = pd.read\_csv('city\_day\_AQI\_test.csv') 9. del dataset\_train['Date'] 10. del dataset\_test['Date'] 11. dataset\_train.fillna(dataset\_train.groupby('City').transform('mean'),inplace=True) 12. dataset\_test.fillna(dataset\_test.groupby('City').transform('mean'),inplace=True) 13. dataset\_train = dataset\_train.fillna(dataset\_train.mean()) 14. dataset\_test = dataset\_test.fillna(dataset\_test.mean()) 15. X = dataset\_train.iloc[:, :-2].values 16. Y = dataset\_train.iloc[:, -2].values 17. X\_test = dataset\_test.iloc[:,:-2].values 18. Y\_test = dataset\_test.iloc[:, -2].values 19. a=X\_test 20. ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0])], remainder='passthrough') 21. X = np.array(ct.fit\_transform(X)) 22. X\_test = np.array(ct.transform(X\_test)) 23. from sklearn.preprocessing import PolynomialFeatures 24. poly\_transform = PolynomialFeatures(degree=2) 25. X\_poly = poly\_transform.fit\_transform(X) 26. X\_poly\_test = poly\_transform.transform(X\_test) 27. poly\_lin\_reg = LinearRegression() 28. poly\_lin\_reg.fit(X\_poly, Y) 29. from sklearn.metrics import mean\_squared\_error 30. from sklearn.metrics import r2\_score 31. from math import sqrt 32. Y\_poly\_pred\_test = poly\_lin\_reg.predict(X\_poly\_test) 33. print("SSE",len(X\_test)\*mean\_squared\_error(Y\_test, Y\_poly\_pred\_test)) 34. print("RMSE", sqrt(mean\_squared\_error(Y\_test, Y\_poly\_pred\_test))) 35. r2=r2\_score(Y\_test, Y\_poly\_pred\_test) 36. print("r2=",r2) 37. adjusted\_r\_squared = 1 - (1-r2)\*((len(Y\_test)-1)/(len(Y\_test)-a.shape[1]-1)) 38. print("adjusted\_r\_squared= ",adjusted\_r\_squared) |

1. **Support Vector Regression (SVR)**

|  |
| --- |
| 1. import pandas as pd 2. import numpy as np 3. from sklearn.preprocessing import OneHotEncoder 4. from sklearn.compose import ColumnTransformer 5. # dataset = pd.read\_csv('Position\_Salaries.csv') 6. dataset\_train = pd.read\_csv('city\_day\_AQI\_train.csv') 7. dataset\_test = pd.read\_csv('city\_day\_AQI\_test.csv') 8. del dataset\_train['Date'] 9. del dataset\_test['Date'] 10. dataset\_train.fillna(dataset\_train.groupby('City').transform('mean'),inplace=True) 11. dataset\_test.fillna(dataset\_test.groupby('City').transform('mean'),inplace=True) 12. dataset\_train = dataset\_train.fillna(dataset\_train.mean()) 13. dataset\_test = dataset\_test.fillna(dataset\_test.mean()) 14. X = dataset\_train.iloc[:, :-2].values 15. Y = dataset\_train.iloc[:, -2].values 16. X\_test = dataset\_test.iloc[:,:-2].values 17. Y\_test = dataset\_test.iloc[:, -2].values 18. a=X\_test 19. ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0])], remainder='passthrough') 20. X = np.array(ct.fit\_transform(X)) 21. X\_test = np.array(ct.transform(X\_test)) 22. Y = Y.reshape(-1,1) 23. Y\_test = Y\_test.reshape(-1,1) 24. from sklearn.preprocessing import StandardScaler 25. sc\_X = StandardScaler() 26. sc\_y = StandardScaler() 27. X\_trans = sc\_X.fit\_transform(X) 28. Y\_trans = sc\_y.fit\_transform(Y) 29. X\_trans\_test = sc\_X.transform(X\_test) 30. Y\_trans\_test = sc\_y.transform(Y\_test) 31. from sklearn.svm import SVR 32. regressor = SVR(kernel = 'rbf') 33. regressor.fit(X\_trans, Y\_trans) 34. def predict(model, X, SC\_X, SC\_Y): 35. X\_trans = SC\_X.transform(X) 36. Y\_trans\_pred = model.predict(X\_trans) 37. Y\_pred = SC\_Y.inverse\_transform(Y\_trans\_pred) 38. return Y\_pred 39. Y\_pred\_train = predict(regressor, X, sc\_X, sc\_y) 40. Y\_pred\_test = predict(regressor, X\_test, sc\_X, sc\_y) 41. from sklearn.metrics import mean\_squared\_error 42. from sklearn.metrics import r2\_score 43. from math import sqrt 44. print("SSE",len(X\_test)\*mean\_squared\_error(Y\_test,Y\_pred\_test)) 45. print("RMSE", sqrt(mean\_squared\_error(Y\_test, Y\_pred\_test))) 46. r2 = r2\_score(Y\_test, Y\_pred\_test) 47. print("r2=",r2) 48. adjusted\_r\_squared = 1 - (1-r2)\*((len(Y\_test)-1)/(len(Y\_test)-a.shape[1]-1)) 49. print("adjusted\_r\_squared= ",adjusted\_r\_squared) |

1. **Decision Tree Regression**

|  |
| --- |
| 1. import pandas as pd 2. import numpy as np 3. from sklearn.preprocessing import OneHotEncoder 4. from sklearn.compose import ColumnTransformer 5. # dataset = pd.read\_csv('Position\_Salaries.csv') 6. dataset\_train = pd.read\_csv('city\_day\_AQI\_train.csv') 7. dataset\_test = pd.read\_csv('city\_day\_AQI\_test.csv') 8. del dataset\_train['Date'] 9. del dataset\_test['Date'] 10. dataset\_train.fillna(dataset\_train.groupby('City').transform('mean'),inplace=True) 11. dataset\_test.fillna(dataset\_test.groupby('City').transform('mean'),inplace=True) 12. dataset\_train = dataset\_train.fillna(dataset\_train.mean()) 13. dataset\_test = dataset\_test.fillna(dataset\_test.mean()) 14. X = dataset\_train.iloc[:, :-2].values 15. Y = dataset\_train.iloc[:, -2].values 16. X\_test = dataset\_test.iloc[:,:-2].values 17. Y\_test = dataset\_test.iloc[:, -2].values 18. a=X\_test 19. ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0])], remainder='passthrough') 20. X = np.array(ct.fit\_transform(X)) 21. X\_test = np.array(ct.transform(X\_test)) 22. Y = Y.reshape(-1,1) 23. Y\_test = Y\_test.reshape(-1,1) 24. from sklearn.tree import DecisionTreeRegressor 25. regressor = DecisionTreeRegressor() 26. regressor.fit(X, Y) 27. Y\_pred\_test=regressor.predict(X\_test) 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)-a.shape[1]-1)) 36. print("adjusted\_r\_squared= ",adjusted\_r\_squared) |

1. **Random Forest Regression**

|  |
| --- |
| 1. import pandas as pd 2. import numpy as np 3. from sklearn.preprocessing import OneHotEncoder 4. from sklearn.compose import ColumnTransformer 5. # dataset = pd.read\_csv('Position\_Salaries.csv') 6. dataset\_train = pd.read\_csv('city\_day\_AQI\_train.csv') 7. dataset\_test = pd.read\_csv('city\_day\_AQI\_test.csv') 8. del dataset\_train['Date'] 9. del dataset\_test['Date'] 10. dataset\_train.fillna(dataset\_train.groupby('City').transform('mean'),inplace=True) 11. dataset\_test.fillna(dataset\_test.groupby('City').transform('mean'),inplace=True) 12. dataset\_train = dataset\_train.fillna(dataset\_train.mean()) 13. dataset\_test = dataset\_test.fillna(dataset\_test.mean()) 14. X = dataset\_train.iloc[:, :-2].values 15. Y = dataset\_train.iloc[:, -2].values 16. X\_test = dataset\_test.iloc[:,:-2].values 17. Y\_test = dataset\_test.iloc[:, -2].values 18. a=X\_test 19. ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0])], remainder='passthrough') 20. X = np.array(ct.fit\_transform(X)) 21. X\_test = np.array(ct.transform(X\_test)) 22. Y = Y.reshape(-1,1) 23. Y\_test = Y\_test.reshape(-1,1) 24. from sklearn.ensemble import RandomForestRegressor 25. regressor = RandomForestRegressor(n\_estimators = 7,random\_state=0 ) 26. regressor.fit(X, Y) 27. Y\_pred\_test=regressor.predict(X\_test) 28. from sklearn.metrics import mean\_squared\_error 29. from sklearn.metrics import r2\_score 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. 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)-a.shape[1]-1)) 36. print("adjusted\_r\_squared= ",adjusted\_r\_squared) |

# Kết quả

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Squared Sum (SSE** | **RMSE** | **R^2** | **R\_adjusted^2** |
| **Linear**  **Regression** | **26800886.85** | **55** | **0.838938357454** | **0.838701662751** |
| **Polynomial**  **Regression** | **18739039.877** | **46** | **0.887386542** | **0.88722104** |
| **Support**  **Vector**  **Regression**  **(SVR)** | **19250528.9379** | **46.613** | **0.8843127159** | **0.88414270** |
| **Decision**  **Tree**  **Regression** | **35066732.743865** | **63** | **0.789264** | **0.788954532** |
| **Random**  **Forest**  **Regression** | **22161420.8821** | **50** | **0.8668195** | **0.8666238** |

Nhận xét:

* Với Hàm lỗi Squared Sum (SSE) thì phương pháp Polynomial Regression là tốt nhất.
* Với Hàm lỗi Root Mean Squared (RMSE) thì phương pháp Polynomial Regression là tốt nhất.
* Với Hàm đánh giá R^2 thì phương pháp Polynomial Regression là tốt nhất.
* Với Hàm đánh giá R\_adjusted^2 thì phương pháp Polynomial Regression là tốt nhất.
* Với tập dữ liệu và test trên thì với phương pháp Polynomial Regression thì cho tất cả các đánh giá độ đo 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 Random Forest Regression không có random\_state = 0 thì sẽ cho những kết quả khác nhau sau mỗi lần chạy.