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.
7. # dataset = pd.read_csv('Position_Salaries.csv')
8. dataset_train = pd.read_csv('city_day_AQI_train.csv')
9. dataset_test = pd.read_csv('city_day_AQI_test.csv')
10. del dataset_train['Date']
11. del dataset_test['Date']
12. dataset_train.fillna(dataset_train.groupby('City').transform('mean'),inplace=Tr
    ue)
13. dataset_test.fillna(dataset_test.groupby('City').transform('mean'),inplace=True
    )
14. dataset_train = dataset_train.fillna(dataset_train.mean())
15. dataset_test = dataset_test.fillna(dataset_test.mean())
16. X = dataset_train.iloc[:, :-2].values
17. Y = dataset_train.iloc[:, -2].values
18. X_test = dataset_test.iloc[:,:-2].values
19. Y_test = dataset_test.iloc[:, -2].values
20. a=X_test
21. ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0])],
    remainder='passthrough')
22. X = np.array(ct.fit_transform(X))
23. X_test = np.array(ct.transform(X_test))
24.
25. lin_reg = LinearRegression()
```

```
26. lin_reg.fit(X, Y)
27. Y_test_pred = lin_reg.predict(X_test)
28.
29. from sklearn.metrics import mean_squared_error
30. from sklearn.metrics import r2_score
31. from math import sqrt
32. print("SSE",len(X_test)*mean_squared_error(Y_test,Y_test_pred))
33. print("RMSE", sqrt(mean_squared_error(Y_test, Y_test_pred)))
34. r2 = r2_score(Y_test, Y_test_pred)
35. print("r2=",r2)
36. adjusted_r_squared = 1 - (1-r2)*((len(Y_test)-1)/(len(Y_test)-a.shape[1]-1))
37. print("adjusted_r_squared= ",adjusted_r_squared)
```

2. **Polynomial Regression**

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer

# dataset = pd.read_csv('Position_Salaries.csv')
dataset_train = pd.read_csv('city_day_AQI_train.csv')
dataset_test = pd.read_csv('city_day_AQI_test.csv')
del dataset_train['Date']
del dataset_test['Date']
dataset_train.fillna(dataset_train.groupby('City').transform('mean'),inplace=True)
dataset_test.fillna(dataset_test.groupby('City').transform('mean'),inplace=True)
dataset_train = dataset_train.fillna(dataset_train.mean())
dataset_test = dataset_test.fillna(dataset_test.mean())
X = dataset_train.iloc[:, :-2].values
Y = dataset_train.iloc[:, -2].values
X_test = dataset_test.iloc[:,:-2].values
Y_test = dataset_test.iloc[:, -2].values
a=X_test
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0])],
    remainder='passthrough')
X = np.array(ct.fit_transform(X))
X_test = np.array(ct.transform(X_test))

from sklearn.preprocessing import PolynomialFeatures
poly_transform = PolynomialFeatures(degree=2)
X_poly = poly_transform.fit_transform(X)
X_poly_test = poly_transform.transform(X_test)
poly_lin_reg = LinearRegression()
poly_lin_reg.fit(X_poly, Y)
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from math import sqrt
Y_poly_pred_test = poly_lin_reg.predict(X_poly_test)
print("SSE",len(X_test)*mean_squared_error(Y_test, Y_poly_pred_test))
print("RMSE", sqrt(mean_squared_error(Y_test, Y_poly_pred_test)))
r2=r2_score(Y_test, Y_poly_pred_test)
print("r2=",r2)
adjusted_r_squared = 1 - (1-r2)*((len(Y_test)-1)/(len(Y_test)-a.shape[1]-1))
print("adjusted_r_squared= ",adjusted_r_squared)
```

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

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer

# dataset = pd.read_csv('Position_Salaries.csv')
dataset_train = pd.read_csv('city_day_AQI_train.csv')
dataset_test = pd.read_csv('city_day_AQI_test.csv')
del dataset_train['Date']
del dataset_test['Date']
dataset_train.fillna(dataset_train.groupby('City').transform('mean'),inplace=True)
dataset_test.fillna(dataset_test.groupby('City').transform('mean'),inplace=True)
dataset_train = dataset_train.fillna(dataset_train.mean())
dataset_test = dataset_test.fillna(dataset_test.mean())
X = dataset_train.iloc[:, :-2].values
Y = dataset_train.iloc[:, -2].values
X_test = dataset_test.iloc[:,:-2].values
Y_test = dataset_test.iloc[:, -2].values
a=X_test
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0])],
remainder='passthrough')
X = np.array(ct.fit_transform(X))
X_test = np.array(ct.transform(X_test))
Y = Y.reshape(-1,1)
Y_test = Y_test.reshape(-1,1)
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
sc_y = StandardScaler()
X_trans = sc_X.fit_transform(X)
Y_trans = sc_y.fit_transform(Y)
X_trans_test = sc_X.transform(X_test)
Y_trans_test = sc_y.transform(Y_test)
from sklearn.svm import SVR
regressor = SVR(kernel = 'rbf')
regressor.fit(X_trans, Y_trans)
def predict(model, X, SC_X, SC_Y):
    X_trans = SC_X.transform(X)
    Y_trans_pred = model.predict(X_trans)
```

```
    Y_pred = SC_Y.inverse_transform(Y_trans_pred)
    return Y_pred
Y_pred_train = predict(regressor, X, sc_X, sc_y)
Y_pred_test = predict(regressor, X_test, sc_X, sc_y)
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from math import sqrt
print("SSE",len(X_test)*mean_squared_error(Y_test,Y_pred_test))
print("RMSE", sqrt(mean_squared_error(Y_test, Y_pred_test)))
r2 = r2_score(Y_test, Y_pred_test)
print("r2=",r2)
adjusted_r_squared = 1 - (1-r2)*((len(Y_test)-1)/(len(Y_test)-a.shape[1]-1))
print("adjusted_r_squared= ",adjusted_r_squared)
```

4. **Decision Tree Regression**

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer

# dataset = pd.read_csv('Position_Salaries.csv')
dataset_train = pd.read_csv('city_day_AQI_train.csv')
dataset_test = pd.read_csv('city_day_AQI_test.csv')
del dataset_train['Date']
del dataset_test['Date']
dataset_train.fillna(dataset_train.groupby('City').transform('mean'),inplace=True)
dataset_test.fillna(dataset_test.groupby('City').transform('mean'),inplace=True)
dataset_train = dataset_train.fillna(dataset_train.mean())
dataset_test = dataset_test.fillna(dataset_test.mean())
X = dataset_train.iloc[:, :-2].values
Y = dataset_train.iloc[:, -2].values
X_test = dataset_test.iloc[:,:-2].values
Y_test = dataset_test.iloc[:, -2].values
a=X_test
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0])],
remainder='passthrough')
X = np.array(ct.fit_transform(X))
X_test = np.array(ct.transform(X_test))
Y = Y.reshape(-1,1)
Y_test = Y_test.reshape(-1,1)
from sklearn.tree import DecisionTreeRegressor
```

```
regressor = DecisionTreeRegressor()
regressor.fit(X, Y)
Y_pred_test=regressor.predict(X_test)

from sklearn.metrics import mean_squared_error
from math import sqrt
print("SSE",len(X_test)*mean_squared_error(Y_test, Y_pred_test))
print("RMSE", sqrt(mean_squared_error(Y_test, Y_pred_test)))
from sklearn.metrics import r2_score
r2=r2_score(Y_test, Y_pred_test)
print("r2=",r2)
adjusted_r_squared = 1 - (1-r2)*((len(Y_test)-1)/(len(Y_test)-a.shape[1]-1))
print("adjusted_r_squared= ",adjusted_r_squared)
```

5. **Random Forest Regression**

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer

# dataset = pd.read_csv('Position_Salaries.csv')
dataset_train = pd.read_csv('city_day_AQI_train.csv')
dataset_test = pd.read_csv('city_day_AQI_test.csv')
del dataset_train['Date']
del dataset_test['Date']
dataset_train.fillna(dataset_train.groupby('City').transform('mean'),inplace=True)
dataset_test.fillna(dataset_test.groupby('City').transform('mean'),inplace=True)
dataset_train = dataset_train.fillna(dataset_train.mean())
dataset_test = dataset_test.fillna(dataset_test.mean())
X = dataset_train.iloc[:, :-2].values
Y = dataset_train.iloc[:, -2].values
X_test = dataset_test.iloc[:,:-2].values
Y_test = dataset_test.iloc[:, -2].values
a=X_test
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0])],
remainder='passthrough')
X = np.array(ct.fit_transform(X))
X_test = np.array(ct.transform(X_test))
Y = Y.reshape(-1,1)
Y_test = Y_test.reshape(-1,1)
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 7,random_state=0  )
```

```
regressor.fit(X, Y)
Y_pred_test=regressor.predict(X_test)
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from math import sqrt
print("SSE",len(X_test)*mean_squared_error(Y_test,Y_pred_test))
print("RMSE", sqrt(mean_squared_error(Y_test, Y_pred_test)))
r2 = r2_score(Y_test, Y_pred_test)
print("r2=",r2)
adjusted_r_squared = 1 - (1-r2)*((len(Y_test)-1)/(len(Y_test)-a.shape[1]-1))
print("adjusted_r_squared= ",adjusted_r_squared)
```

3. 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.