BÀI TUẦN 04: EVALUATING REGRESSION MODELS PERFORMANCE

1.Thông tin sinh viên

DƯƠNG MINH LƯỢNG-18521071

2.Source

1. Linear Regression

```
import matplotlib.pyplot as plt
import pandas as pd

# dataset = pd.read_csv('Position_Salaries.csv')
dataset_train = pd.read_csv('Position_SalariesTrain.csv')
dataset_test = pd.read_csv('Position_SalariesTest.csv')
X = dataset_train.iloc[:, 1:-1].values
Y = dataset_train.iloc[:, -1].values
X_test = dataset_test.iloc[:, 1:-1].values
Y_test = dataset_test.iloc[:, -1].values

from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X, Y)
Y_pred = lin_reg.predict(X)
Y_pred_test = lin_reg.predict(X_test)
plt.scatter(X_test, Y_test, color = "red")
plt.scatter(X_test, Y_pred_test, color = "black")
plt.plot(X, Y_pred, color = "blue")
plt.title("Position Level vs Salary (Linear Regression)")
plt.xlabel("Position Level")
plt.ylabel("Salary (dollars/year)")
plt.show()

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)-X_test.shape[1]-1))
print("adjusted_r_squared",adjusted_r_squared)
import statsmodels.regression.linear_model as sm
regressor_OLS = sm.OLS(endog = Y, exog = X).fit()
print(regressor_OLS.summary())
```

2. Polynomial Regression

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
# dataset = pd.read_csv('Position_Salaries.csv')
dataset_train = pd.read_csv('Position_SalariesTrain.csv')
dataset_test = pd.read_csv('Position_SalariesTest.csv')
X = dataset_train.iloc[:, 1:-1].values
Y = dataset_train.iloc[:, -1].values
X_test = dataset_test.iloc[:, 1:-1].values
Y_test = dataset_test.iloc[:, -1].values
from sklearn.preprocessing import PolynomialFeatures
poly_transform = PolynomialFeatures(degree=4)
X_poly = poly_transform.fit_transform(X)
X_poly_test = poly_transform.fit_transform(X_test)
from sklearn.linear_model import LinearRegression
poly_lin_reg = LinearRegression()
poly_lin_reg.fit(X_poly, Y)
Y_poly_pred = poly_lin_reg.predict(X_poly)
Y_poly_pred_test = poly_lin_reg.predict(X_poly_test)
plt.scatter(X_test, Y_test ,color = "red")
plt.plot(X, Y_poly_pred, color = "blue")
plt.scatter(X_test, Y_poly_pred_test ,color = "black")
plt.title("Position Level vs Salary(Polynomial Regression)")
plt.xlabel("Position Level")
plt.ylabel("Salary (dollars/year)")
plt.show()
X_grid = np.arange(0, max(X)+1, 0.1)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(X_test, Y_test, color = "red")
plt.plot(X_grid, poly_lin_reg.predict(poly_transform.fit_transform(X_grid)), color = 'blue')
plt.scatter(X_test, Y_poly_pred_test, color = "black")
```

```
plt.title("Position Level vs Salary(Polynomial Regression)")
plt.xlabel("Position Level")
plt.ylabel("Salary (dollars/year)")
plt.show()


# from sklearn.metrics import mean_absolute_error
# print("MAE", mean_absolute_error(Y_test, Y_poly_pred_test))
from sklearn.metrics import mean_squared_error
from math import sqrt
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)))
from sklearn.metrics import r2_score
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)-X_test.shape[1]-1))
print("adjusted_r_squared= ",adjusted_r_squared)
import statsmodels.regression.linear_model as sm
regressor_OLS = sm.OLS(endog = Y, exog = X).fit()
print(regressor_OLS.summary())
```

3. Support Vector Regression (SVR)

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# dataset = pd.read_csv('Position_Salaries.csv')
dataset_train = pd.read_csv('Position_SalariesTrain.csv')
dataset_test = pd.read_csv('Position_SalariesTest.csv')
X = dataset_train.iloc[:, 1:-1].values
Y = dataset_train.iloc[:, -1].values
X_test = dataset_test.iloc[:, 1:-1].values
Y_test = dataset_test.iloc[:, -1].values
Y = Y.reshape(len(Y),1)
Y_test = Y_test.reshape(len(Y_test),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)
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)
plt.scatter(X_test, Y_test, color = 'red')
plt.scatter(X_test, Y_pred_test, color = 'black')
plt.plot(X, Y_pred_train, color = 'blue')
plt.title('Truth or Bluff (SVR)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()

X_grid = np.arange(0, 11, 0.1)
X_grid= X_grid.reshape((len(X_grid), 1))
Y_pred_grid = predict(regressor, X_grid, sc_X, sc_y)
plt.scatter(X_test, Y_test, color = 'red')
plt.scatter(X_test,  Y_pred_test, color = 'black')
plt.plot(X_grid, Y_pred_grid, color = 'blue')
plt.title('Truth or Bluff (SVR)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()

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)-X_test.shape[1]-1))
print("adjusted_r_squared= ",adjusted_r_squared)
```

4. Decision Tree Regression

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# dataset = pd.read_csv('Position_Salaries.csv')
dataset_train = pd.read_csv('Position_SalariesTrain.csv')
dataset_test = pd.read_csv('Position_SalariesTest.csv')
```

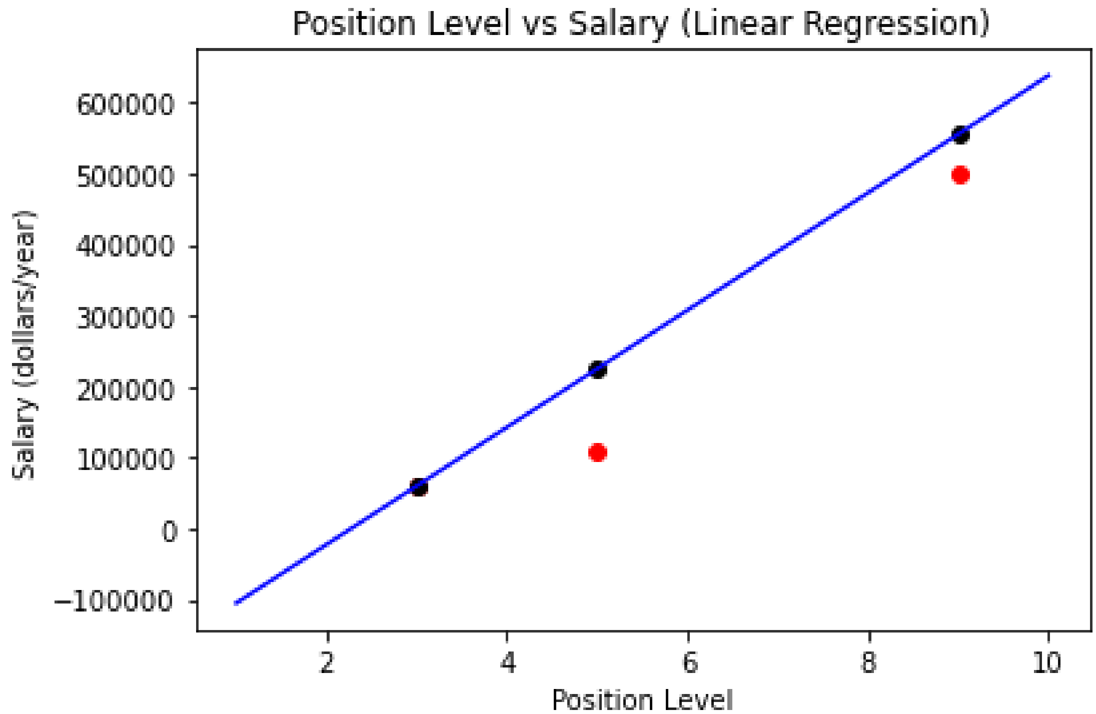
```
X = dataset_train.iloc[:, 1:-1].values
Y = dataset_train.iloc[:, -1].values
X_test = dataset_test.iloc[:, 1:-1].values
Y_test = dataset_test.iloc[:, -1].values
Y = Y.reshape(len(Y),1)
Y_test = Y_test.reshape(len(Y_test),1)
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(X, Y)
Y_pred_test=regressor.predict(X_test)
X_grid = np.arange(min(X), max(X), 0.01)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(X_test, Y_test, color = 'red')
plt.scatter(X_test, Y_pred_test, color = 'black')
plt.plot(X_grid, regressor.predict(X_grid), color = 'blue')
plt.title('Truth or Bluff (Decision Tree Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()

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)-X_test.shape[1]-1))
print("adjusted_r_squared= ",adjusted_r_squared)
```

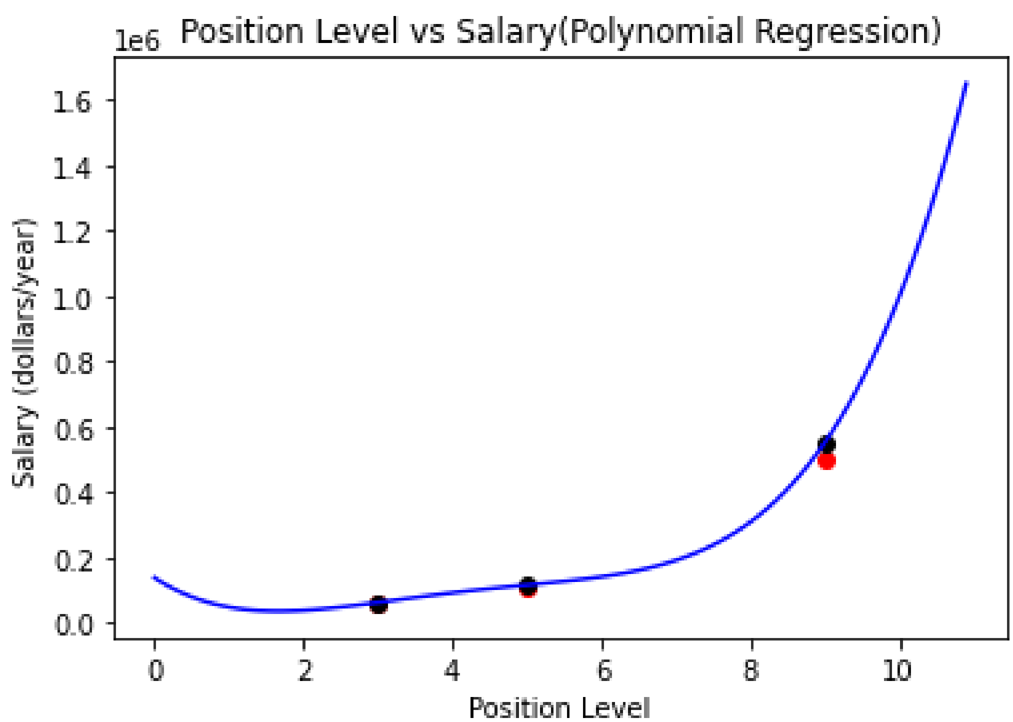
5. Random Forest Regression

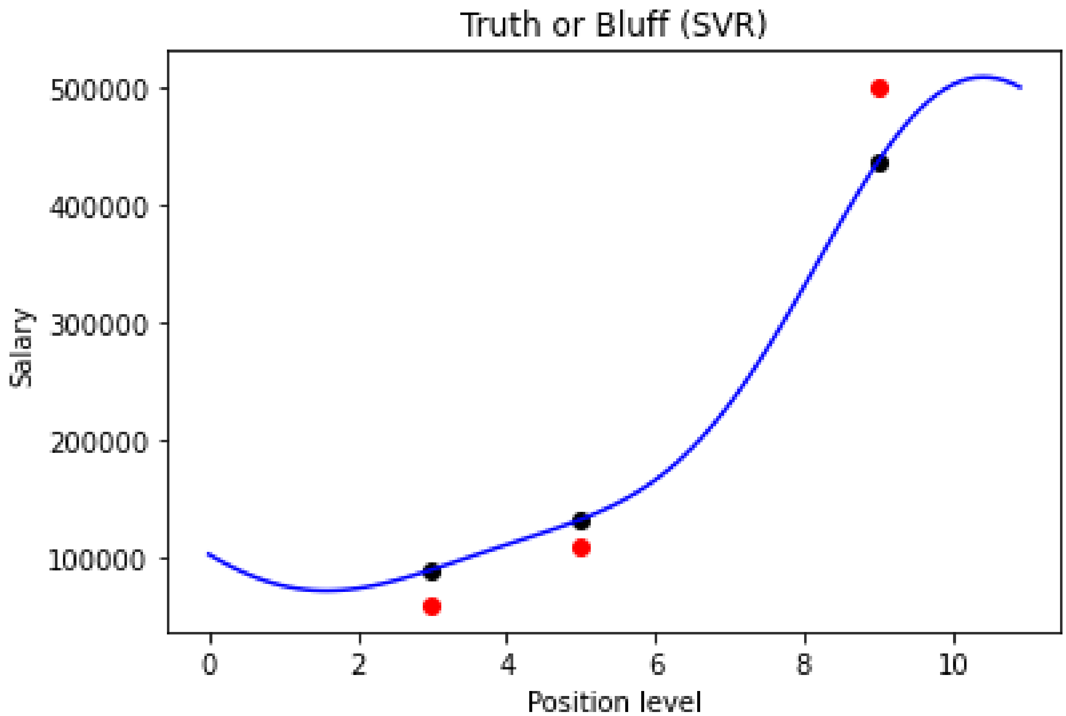
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# dataset = pd.read_csv('Position_Salaries.csv')
```

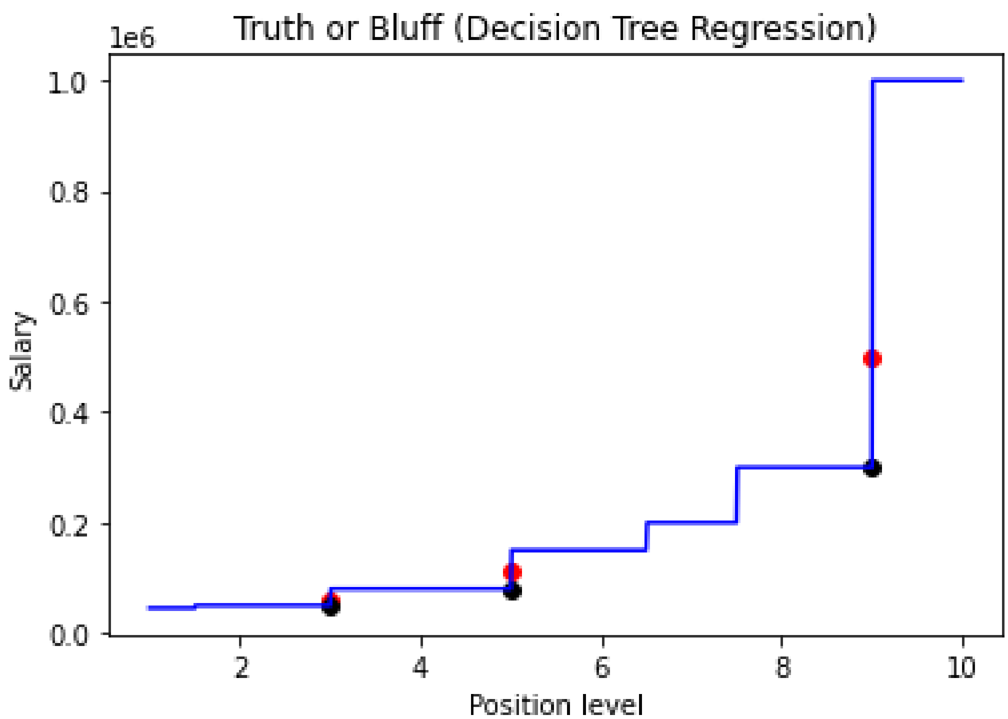
```
dataset_train = pd.read_csv('Position_SalariesTrain.csv')
dataset_test = pd.read_csv('Position_SalariesTest.csv')
X = dataset_train.iloc[:, 1:-1].values
Y = dataset_train.iloc[:, -1].values
X_test = dataset_test.iloc[:, 1:-1].values
Y_test = dataset_test.iloc[:, -1].values
Y = Y.reshape(len(Y),1)
Y_test = Y_test.reshape(len(Y_test),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)
X_grid = np.arange(min(X), max(X), 0.1)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(X_test, Y_test, color = 'red')
plt.scatter(X_test, Y_pred_test, color = 'black')
plt.plot(X_grid, regressor.predict(X_grid), color = 'blue')
plt.title('Truth or Bluff (Random Forest Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()

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)-X_test.shape[1]-1))
print("adjusted_r_squared= ",adjusted_r_squared)
```

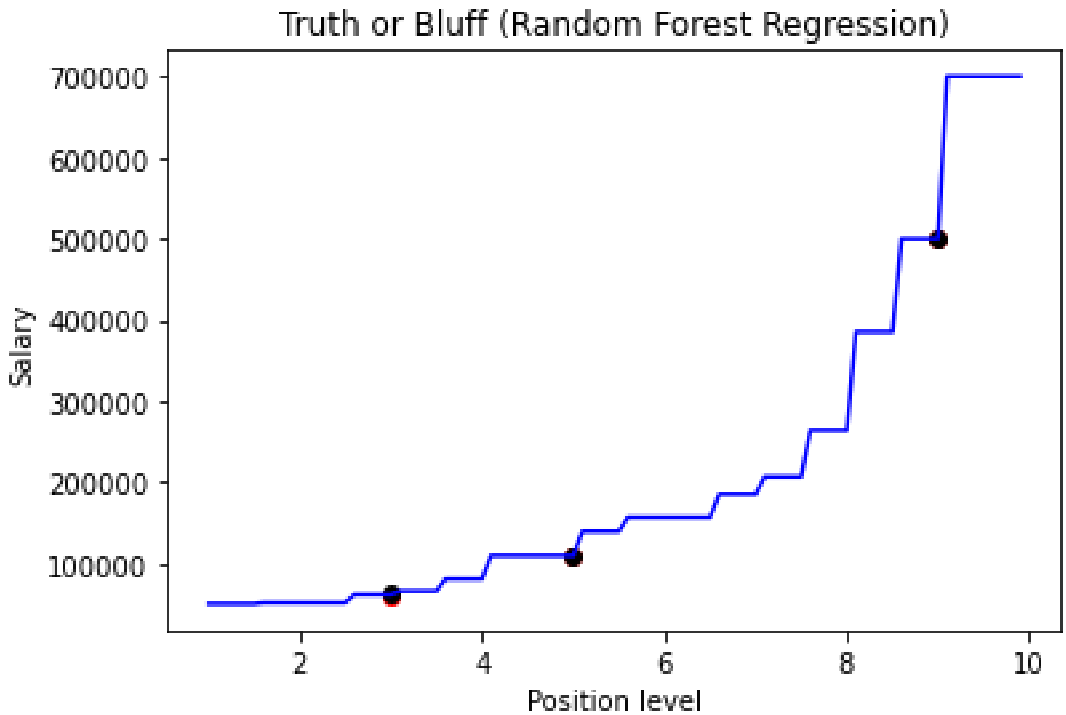
3. Kết quả


Position Level vs Salary (Linear Regression)
600000
500000
400000
300000
200000
100000
0
−100000
Salary (dollars/year)
2
4
6
8
10
Position Level


Position Level vs Salary(Polynomial Regression)
1e6
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Salary (dollars/year)
0
2
4
6
8
10
Position Level


Truth or Bluff (SVR)
500000
400000
300000
200000
100000
Salary
0
2
4
6
8
10
Position level


Truth or Bluff (Decision Tree Regression)
1e6
1.0
0.8
0.6
0.4
0.2
0.0
Salary
2
4
6
8
10
Position level



	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	**4591836.734693887**	**1237.179148263485**	**0.999960437937380**	**0.9999208758747611**

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_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.