Regresi

Oleh: Kelompok GCD

211110347 - Cindy Sintiya

211111930 - David Bate'e

211110948 - Grace Helena Hutagaol

Prediksi Harga Penjualan Mobil dengan Dataset "Automobile Dataset" Menggunakan Metode Regresi

Deskripsi Dataset

Sumber dataset:
kaggle.com/datasets/toramky/automobile-dataset

01.

Ukuran Data (row, col):

```
raw_data.shape
```

```
(205, 26)
```

02.

Preview:

```
raw_data = pd.read_csv("Automobile_data.csv")
raw_data.head()
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg	price
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	13495
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	16500
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000	19	26	16500
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.4	10.0	102	5500	24	30	13950
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.4	8.0	115	5500	18	22	17450

5 rows × 26 columns

Fitur-Fitur Dataset

Mentah

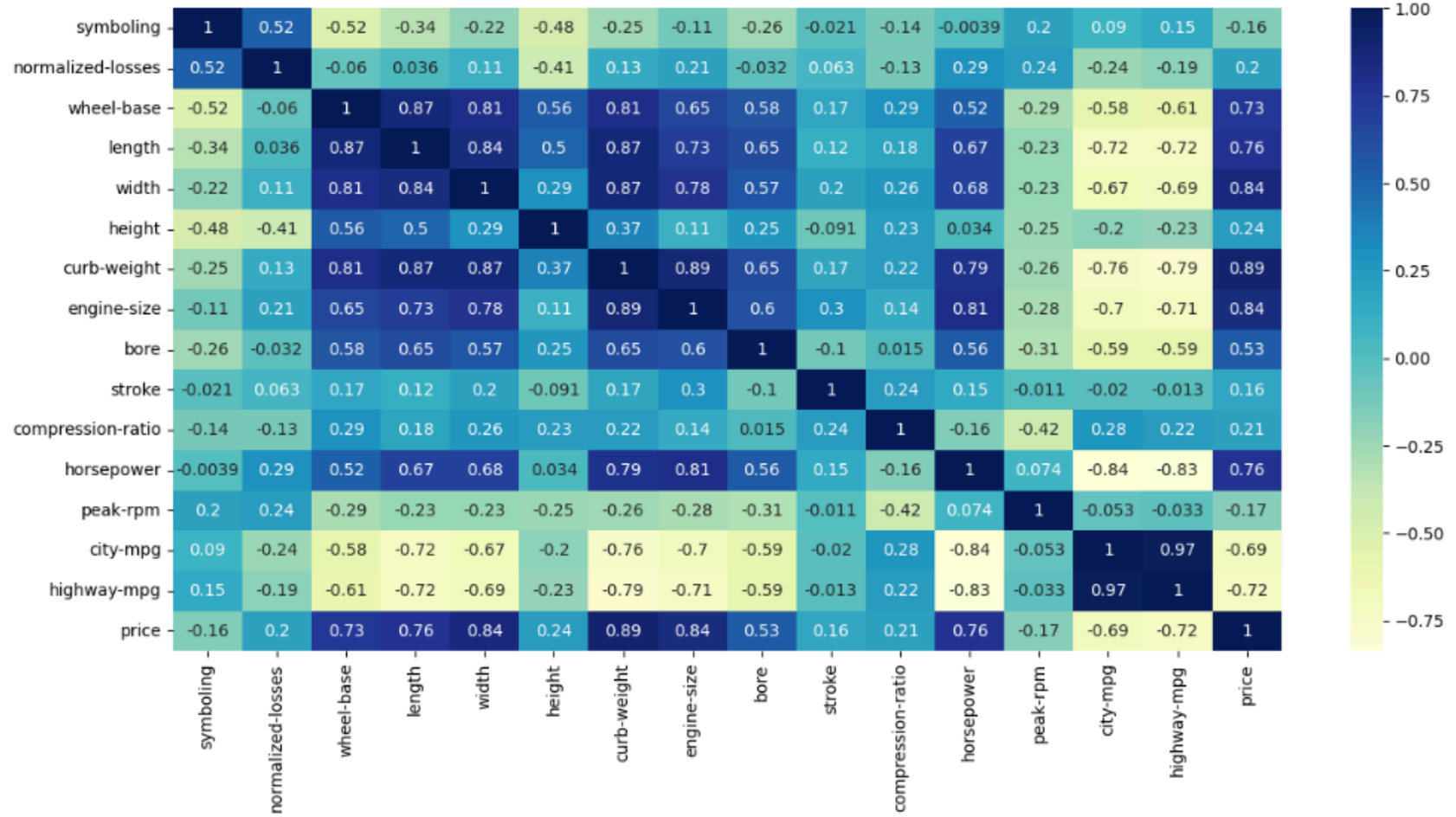
```
[ ]  raw_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   symboling          205 non-null    int64
 1   normalized-losses  205 non-null    object
 2   make               205 non-null    object
 3   fuel-type          205 non-null    object
 4   aspiration         205 non-null    object
 5   num-of-doors       205 non-null    object
 6   body-style         205 non-null    object
 7   drive-wheels       205 non-null    object
 8   engine-location    205 non-null    object
 9   wheel-base         205 non-null    float64
 10  length             205 non-null    float64
 11  width              205 non-null    float64
 12  height             205 non-null    float64
 13  curb-weight        205 non-null    int64
 14  engine-type        205 non-null    object
 15  num-of-cylinders   205 non-null    object
 16  engine-size        205 non-null    int64
 17  fuel-system        205 non-null    object
 18  bore               205 non-null    object
 19  stroke             205 non-null    object
 20  compression-ratio  205 non-null    float64
 21  horsepower         205 non-null    object
 22  peak-rpm           205 non-null    object
 23  city-mpg           205 non-null    int64
 24  highway-mpg        205 non-null    int64
 25  price              205 non-null    object
dtypes: float64(5), int64(5), object(16)
memory usage: 41.8+ KB
```

Korelasi antar Variabel (Numerik)

```
fig, ax = plt.subplots(figsize=(15, 7))
sns.heatmap(raw_data.corr(numeric_only=True), cmap="YlGnBu", annot=True)
```

```
<Axes: >
```



Preprocessing Dataset

Data Cleaning

01.

02.

03.

```
[5]  # delete row that contain any "?"

     for i in raw_data.columns:
       raw_data = raw_data[raw_data[i] != '?']

     print(raw_data.shape)
```

```
(159, 26)
```

```
# transform string numeric data to integer

for i in raw_data.columns:
  if raw_data[i].dtype == 'object':
    try:
      if "." in str(raw_data[i]):
        raw_data[i] = raw_data[i].astype('float64')
      else:
        raw_data[i] = raw_data[i].astype('int64')
    except:
      continue

raw_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 159 entries, 3 to 204
Data columns (total 26 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   symboling          159 non-null    int64
 1   normalized-losses  159 non-null    float64
 2   make               159 non-null    object
 3   fuel-type          159 non-null    object
 4   aspiration         159 non-null    object
 5   num-of-doors       159 non-null    object
 6   body-style         159 non-null    object
 7   drive-wheels       159 non-null    object
 8   engine-location    159 non-null    object
 9   wheel-base         159 non-null    float64
 10  length             159 non-null    float64
 11  width              159 non-null    float64
 12  height             159 non-null    float64
 13  curb-weight        159 non-null    int64
 14  engine-type        159 non-null    object
 15  num-of-cylinders   159 non-null    object
 16  engine-size        159 non-null    int64
 17  fuel-system        159 non-null    object
 18  bore               159 non-null    float64
 19  stroke             159 non-null    float64
 20  compression-ratio  159 non-null    float64
 21  horsepower         159 non-null    float64
 22  peak-rpm           159 non-null    float64
 23  city-mpg           159 non-null    int64
 24  highway-mpg        159 non-null    int64
 25  price              159 non-null    float64
dtypes: float64(11), int64(5), object(10)
memory usage: 33.5+ KB
```

```
# remove unused column

new_data = pd.DataFrame({
    "merek": raw_data["make"],
    # "bahan-bakar": raw_data["fuel-type"],
    # "udara": raw_data["aspiration"],
    "jlh-pintu": raw_data["num-of-doors"],
    # "body": raw_data["body-style"],
    # "jenis-ban": raw_data["drive-wheels"],
    "roda": raw_data["wheel-base"],
    # "panjang": raw_data["length"],
    "lebar": raw_data["width"],
    "berat": raw_data["curb-weight"],
    # "jlh-silinder": raw_data["num-of-cylinders"],
    # "tipe-mesin": raw_data["engine-type"],
    "uk-mesin": raw_data["engine-size"],
    "tenaga": raw_data["horsepower"],
    # "rasio-comp": raw_data["compression-ratio"],
    "price": raw_data["price"],
})
new_data.head()
```

	merek	jlh-pintu	roda	lebar	berat	uk-mesin	tenaga	price
3	audi	four	99.8	66.2	2337	109	102.0	13950.0
4	audi	four	99.4	66.4	2824	136	115.0	17450.0
6	audi	four	105.8	71.4	2844	136	110.0	17710.0
8	audi	four	105.8	71.4	3086	131	140.0	23875.0
10	bmw	two	101.2	64.8	2395	108	101.0	16430.0

Train-Test Dataset Splitting

```
[ ]  # memisahkan variabel x dan y dari tabel
     X_data = new_data.drop('price', axis = 1)
     y_data = new_data['price']
```

```
[ ]  X_data = pd.get_dummies(X_data, drop_first = True, dtype = 'int8')
```

```
# bagi data untuk training n testing dgn rasio 8:2

X_train, X_test, y_train, y_test = train_test_split(
    X_data,
    y_data,
    test_size = 0.20,
    random_state = 50
)
```

Modelling Algorithm

01

Linear Regression

02

Polynomial Regression

03

KNN (K-Nearest Neighbors)

04

Regression Tree

Linear Regression

```
linreg_model = LinearRegression()
linreg_model.fit(X_train, y_train)

# membuat prediksi model
y_pred_linreg = linreg_model.predict(X_test)
```

```
[17] # Evaluasi model linear regression - R^2, MSE, RMSE, MAPE
     print("R^2  =", linreg_model.score(X_test,y_test))
     print("MSE  =", metrics.mean_squared_error(y_test, y_pred_linreg))
     print("RMSE =", np.sqrt(metrics.mean_squared_error(y_test, y_pred_linreg)))
     print("MAPE =", metrics.mean_absolute_percentage_error(y_test, y_pred_linreg) * 100, "%")
```

```
R^2  = 0.8552407168364937
MSE  = 3651638.922343088
RMSE = 1910.9261948968851
MAPE = 12.75246799353539 %
```

Polynomial Regression

```
polyreg_model = LinearRegression()
polyreg_model.fit(X_train_poly, y_train)

# membuat prediksi model
y_pred_polyreg = polyreg_model.predict(X_test_poly)
```

```
[21] # Evaluasi model polinomial regression - R^2, MSE, RMSE, MAPE
     print("R^2  =", polyreg_model.score(X_test_poly, y_test))
     print("MSE  =", metrics.mean_squared_error(y_test, y_pred_polyreg))
     print("RMSE =", np.sqrt(metrics.mean_squared_error(y_test, y_pred_polyreg)))
     print("MAPE =", metrics.mean_absolute_percentage_error(y_test, y_pred_polyreg) * 100, "%")
```

```
R^2  = -23839127.868086226
MSE  = 601356189026792.2
RMSE = 24522564.89494507
MAPE = 24248.45574343964 %
```

KNN

```
knn_model = KNeighborsRegressor(n_neighbors=5)
knn_model.fit(X_train, y_train)

# membuat prediksi model
y_pred_knn = knn_model.predict(X_test)
```

```
[23] # Evaluasi model KNN - R^2, MSE, RMSE, MAPE
     print("R^2  =", knn_model.score(X_test, y_test))
     print("MSE  =", metrics.mean_squared_error(y_test, y_pred_knn))
     print("RMSE =", np.sqrt(metrics.mean_squared_error(y_test, y_pred_knn)))
     print("MAPE =", metrics.mean_absolute_percentage_error(y_test, y_pred_knn) * 100, "%")
```

```
R^2  = 0.7271357395208278
MSE  = 6883163.084999999
RMSE = 2623.5782978596235
MAPE = 14.545273801848133 %
```

Regression Tree

```
regtree_model = DecisionTreeRegressor(max_depth=10)
regtree_model.fit(X_train, y_train)

# membuat prediksi model
y_pred_regtree = regtree_model.predict(X_test)
```

```
[32] # Evaluasi model regression tree - R^2, MSE, RMSE, MAPE
     print("R^2  =", regtree_model.score(X_test, y_test))
     print("MSE  =", metrics.mean_squared_error(y_test, y_pred_regtree))
     print("RMSE =", np.sqrt(metrics.mean_squared_error(y_test, y_pred_regtree)))
     print("MAPE =", metrics.mean_absolute_percentage_error(y_test, y_pred_regtree) * 100, "%")
```

```
R^2  = 0.901427271806723
MSE  = 2486555.6328125
RMSE = 1576.8816166131496
MAPE = 9.656007165029706 %
```

Perbandingan Error Model

Tabel Perbandingan Algoritma Terbaik

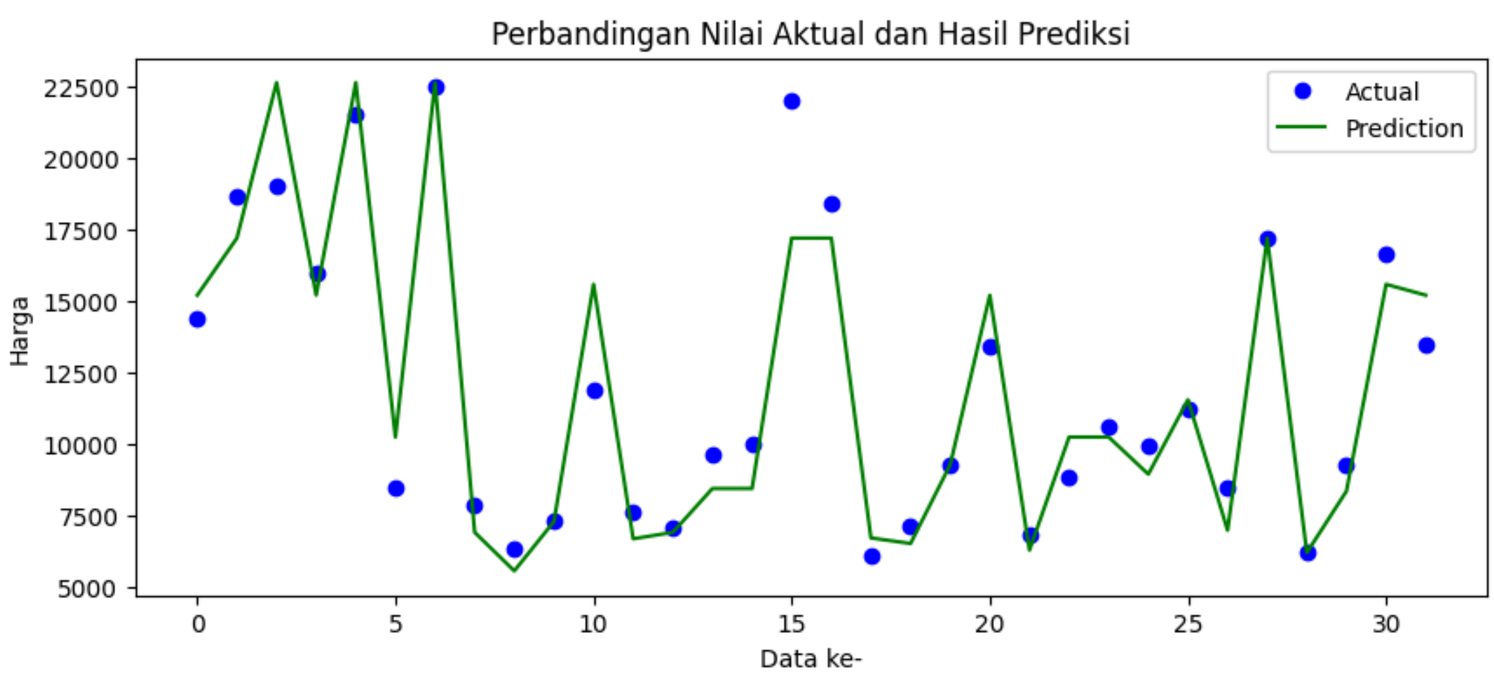
```
algo_result = pd.DataFrame({
    "Algorithm": ["Linear Regression", "Polynomial Regression", "KNN", "Regression Tree"],
    "R^2": [linreg_model.score(X_test, y_test), polyreg_model.score(X_test_poly, y_test), knn_mode
    "MSE": [metrics.mean_squared_error(y_test, linreg_result), metrics.mean_squared_error(y_test,
    "RMSE": [np.sqrt(metrics.mean_squared_error(y_test, linreg_result)), np.sqrt(metrics.mean_squa
    "MAPE": [metrics.mean_absolute_percentage_error(y_test, linreg_result) * 100, metrics.mean_abs
})

algo_result.round(3)
algo_result.style.format({"R^2":"{:,.3f}", "MSE":"{:,.3f}", "RMSE":"{:,.3f}", "MAPE":"{:.3f}%"})
```

	Algorithm	R^2	MSE	RMSE	MAPE
0	Linear Regression	0.855	3,651,638.922	1,910.926	12.752%
1	Polynomial Regression	-23,839,127.868	601,356,189,026,792.250	24,522,564.895	24248.456%
2	KNN	0.727	6,883,163.085	2,623.578	14.545%
3	Regression Tree	0.901	2,486,555.633	1,576.882	9.656%

Kesimpulan

Visualisasi Nilai Aktual dengan Nilai Hasil Prediksi

```
plt.figure(figsize=(10, 4))
plt.plot(np.array(y_test), 'bo', label='Actual')
plt.plot(y_pred_regtree, color="green", label='Prediction')
plt.title('Perbandingan Nilai Aktual dan Hasil Prediksi')
plt.xlabel('Data ke-')
plt.ylabel('Harga')
plt.legend(loc='upper right')
plt.show()
```



Metode Regresi terbaik untuk **prediksi harga mobil** dengan dataset **"Automobile-Dataset"** adalah **Regression Tree** dengan persentase error terkecil yaitu **9.65%** diukur dengan metriks pengujian **MAPE** (Mean Absolute Percentage Error)

DONE!

Akses Coding Disini

- Tim GCD -