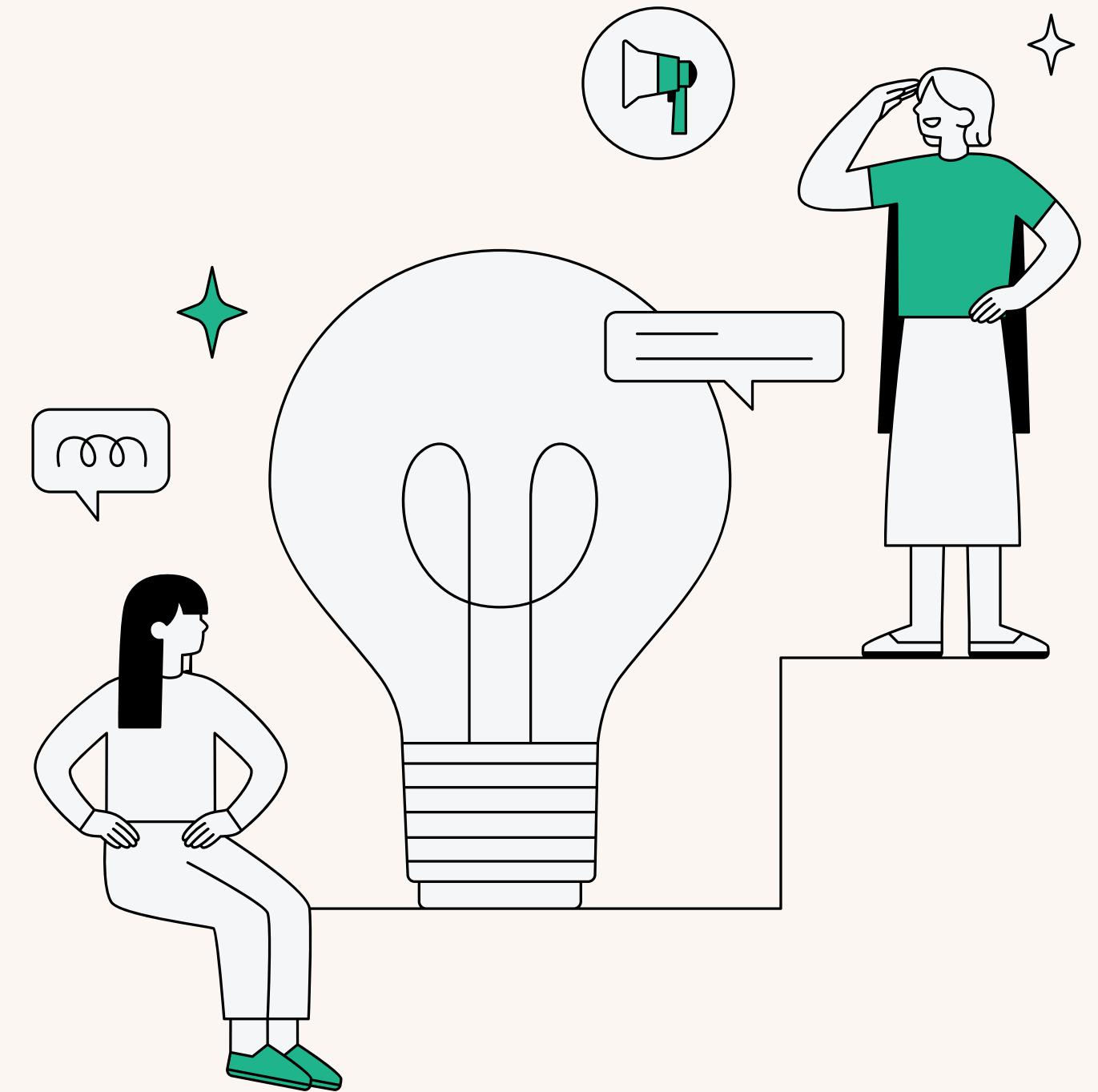


# Pemrograman Learning

IF-46-04

- Muhammad Fakhrul Hizrian (1301223174)
- Muhammad Rifki Hidayatullah (1301220250)



# Data Preparation

```
▼ Import Library

[ ] import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.preprocessing import StandardScaler
```

- pandas dan numpy: Digunakan untuk manipulasi data dan perhitungan numerik.
- seaborn dan matplotlib.pyplot: Digunakan untuk visualisasi data.
- scikit-learn: Menyediakan fungsi untuk pemodelan dan evaluasi model (regresi linear, splitting data, scaling, dan metrik evaluasi).

```
[14] dataset = pd.read_csv("abalone.csv")

[15] print(dataset)

      Sex  Length  Diameter  Height  Whole weight  Shucked weight \
0       M     0.455     0.365    0.095      0.5140      0.2245
1       M     0.350     0.265    0.090      0.2255      0.0995
2       F     0.530     0.420    0.135      0.6770      0.2565
3       M     0.440     0.365    0.125      0.5160      0.2155
4       I     0.330     0.255    0.080      0.2050      0.0895
...     ...
4172    F     0.565     0.450    0.165      0.8870      0.3700
4173    M     0.590     0.440    0.135      0.9660      0.4390
4174    M     0.600     0.475    0.205      1.1760      0.5255
4175    F     0.625     0.485    0.150      1.0945      0.5310
4176    M     0.710     0.555    0.195      1.9485      0.9455

      Viscera weight  Shell weight  Rings
0           0.1010      0.1500     15
1           0.0485      0.0700      7
2           0.1415      0.2100      9
3           0.1140      0.1550     10
4           0.0395      0.0550      7
...         ...
4172        0.2390      0.2490     11
4173        0.2145      0.2605     10
4174        0.2875      0.3080      9
4175        0.2610      0.2960     10
4176        0.3765      0.4950     12

[4177 rows x 9 columns]
```

pd.read\_csv: Membaca dataset dari URL dan mengatur nama kolom.  
Print(dataset): mengeluarkan semua data

# Data Preprocessing

✓ Konversi kolom kategorikal sex menjadi value numerik

```
[ ] dataset['Sex'] = dataset['Sex'].map({'M': 1, 'F': 2, 'I': 3})
```

Mengonversi kolom kategorikal 'Sex' menjadi nilai numerik: M (1), F (2), I (3).

✓ Cek data

```
[ ] dataset.head()
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	1	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	1	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	3	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

Data sudah berhasil diubah

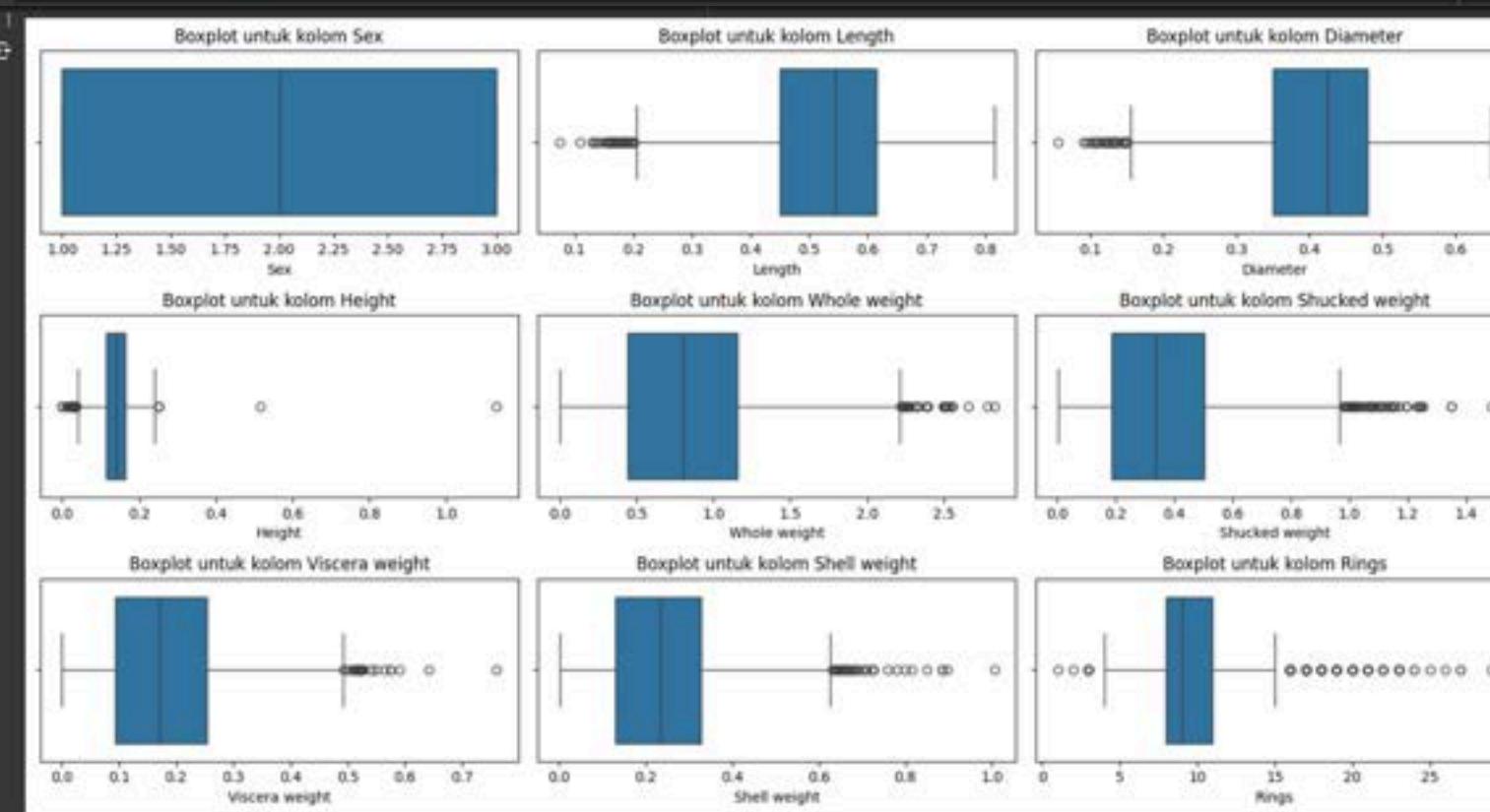
```
[ ] print("\nMissing Values:")
print(dataset.isnull().sum())
```

```
→
Missing Values:
Sex          0
Length       0
Diameter     0
Height        0
Whole weight 0
Shucked weight 0
Viscera weight 0
Shell weight 0
Rings         0
dtype: int64
```

Cek data apakah data tersebut ada yang hilang atau tidak (Null)

### ✓ Mengecek Outlier

```
[ ] # Menampilkan plot sebelum penanganan outliers
num_cols = len(dataset.select_dtypes(include=['int','float']).columns)
plt.figure(figsize=(15,8))
rows = (num_cols + 2) // 3
for i,col in enumerate(dataset.select_dtypes(include=['int','float']).columns):
    plt.subplot(rows, 3, i+1)
    sns.boxplot(x=dataset[col])
    plt.title(f'Boxplot untuk kolom {col}')
plt.tight_layout()
plt.show()
```



### ✓ Menangani Outliers

```
[ ] # Mendeksi dan menangani outlier menggunakan metode IQR
def handle_outliers(data, column):
    Q1 = data[column].quantile(0.25)
    Q3 = data[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    data[column] = data[column].apply(lambda x: upper_bound if x > upper_bound else (lower_bound if x < lower_bound else x))
    return data

for col in dataset.select_dtypes(include=['int','float']).columns:
    dataset = handle_outliers(dataset, col)

print("Outliers berhasil ditangani")
```

Outliers berhasil ditangani

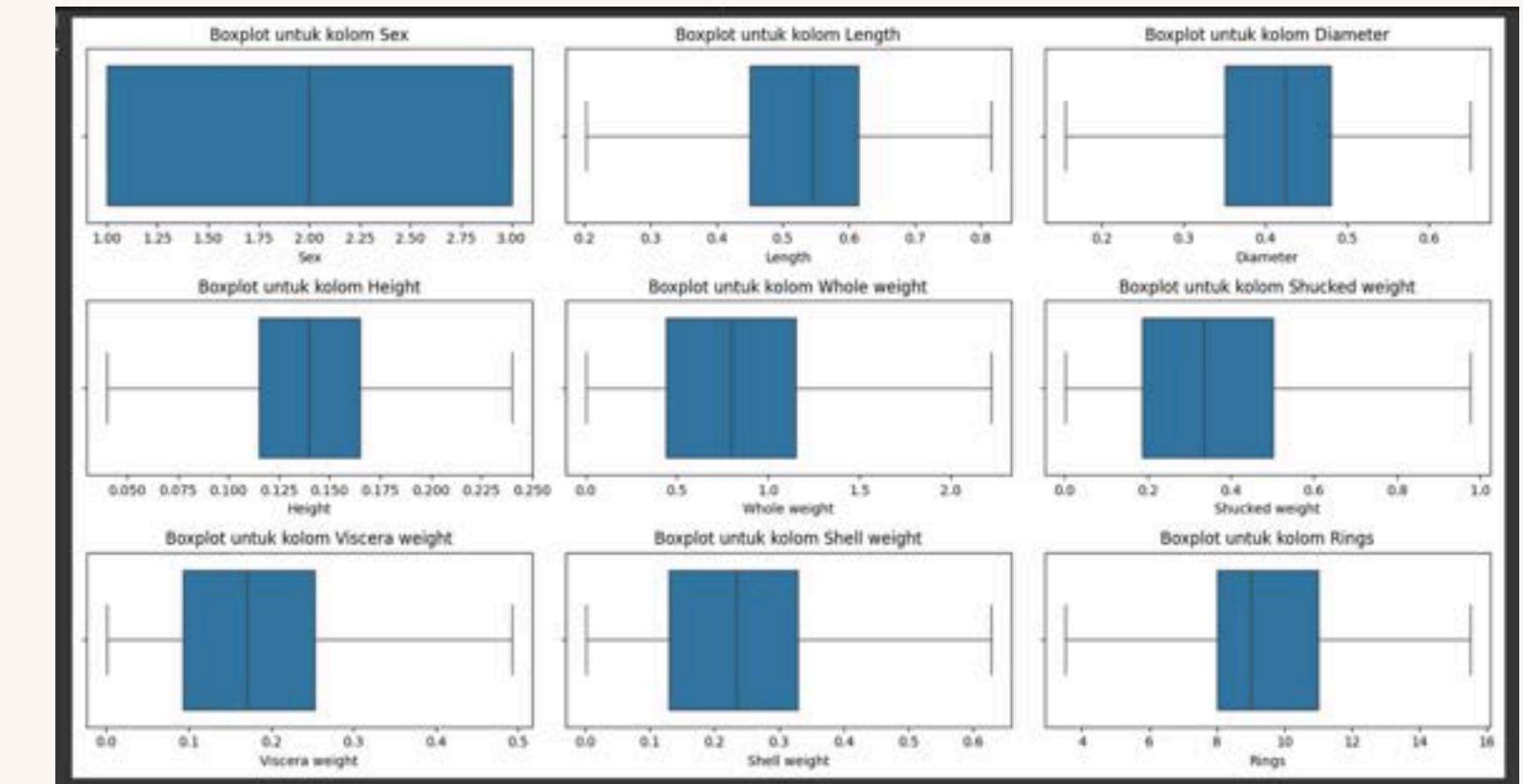
Dan keputusan yang diambil yaitu menghapus outlier untuk meningkatkan akurasi model, terutama yang sensitif terhadap data yang ekstrim seperti regresi linear. Dengan menghapus outlier, model dapat lebih baik mempelajari pola yang ada dalam data dan menghasilkan prediksi yang lebih akurat.

Cek outliers untuk mendeksi kesalahan data dan apakah dalam klasifikasi ini perlu untuk menghapus outlier

### ▼ Mengecek Outlier

```
[ ] # Menampilkan plot sesudah penanganan outliers
num_cols = len(dataset.select_dtypes(include=['int','float']).columns)
plt.figure(figsize=(15,8))
rows = (num_cols + 2) // 3
for i,col in enumerate(dataset.select_dtypes(include=['int','float']).columns):
    plt.subplot(rows, 3, i+1)
    sns.boxplot(x=dataset[col])
    plt.title(f'Boxplot untuk kolom {col}')
plt.tight_layout()
plt.show()
```

Kodingan diatas adalah kodingan yang berfungsi untuk menangani outliers, dan bisa kita lihat hasilnya boxplot yang dihasilkan telah dihilangkan outliers nya



## ✓ Membagi dataset menjadi X dan Y

```
[ ] X = dataset.iloc[:, 2: ].values  
Y = dataset.iloc[:, 1].values
```

### Membagi dataset menjadi X dan Y

- X: Fitur yang digunakan untuk prediksi
- Y: Target yang akan diprediksi

## Menampilkan beberapa data dari X

```
[ ] print("Head of X:")  
print(X[:5, :])
```

→ Head of X:  
[[ 0.365 0.095 0.514 0.2245 0.101 0.15 15. ]  
[ 0.265 0.09 0.2255 0.0995 0.0485 0.07 7. ]  
[ 0.42 0.135 0.677 0.2565 0.1415 0.21 9. ]  
[ 0.365 0.125 0.516 0.2155 0.114 0.155 10. ]  
[ 0.255 0.08 0.205 0.0895 0.0395 0.055 7. ]]

```
[ ] print("Head of Y:")  
print(Y[:5])
```

```
→ Head of Y:  
[0.455 0.35 0.53 0.44 0.33 ]
```

## Menampilkan beberapa data dari Y

### ▼ Memisahkan Dataset Menjadi Data Latih(Training Set) dan Data Uji (Test Set)

membagi dataset menjadi dua bagian, yaitu data pelatihan(X\_train dan Y\_train) dan data pengujian(X\_test dan Y\_test), yang dalam kasus ini, dataset akan dibagi menjadi data pelatihan(75%) dan data pengujian(25%)

```
[ ] from sklearn.model_selection import train_test_split  
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, random_state = 0)
```

Memisahkan Dataset Menjadi data latih dan data uji yaitu data pelatihan(X\_train dan Y\_train) dan data pengujian(X\_test dan Y\_test), yang dalam kasus ini, dataset akan dibagi menjadi data pelatihan(75%) dan data pengujian(25%)

### ▼ Penskalaan Fitur

Melakukan penskalaan fitur sehingga mean sekitar 0 dan simpangan baku (standard deviation) sekitar 1

```
[ ] from sklearn.preprocessing import StandardScaler  
sc = StandardScaler()  
X_train = sc.fit_transform(X_train)  
X_test = sc.transform(X_test)
```

Melakukan penskalaan fitur sehingga mean sekitar 0 dan simpangan baku (standard deviation) sekitar 1 sehingga berfungsi untuk meningkatkan akurasi model dan mengurangi sensitivitas terhadap outlier

```
[ ] print("Head of X_train")  
print(X_train[:5, :])  
  
print("\nHead of X_test:")  
print(X_test[:5, :])
```

Head of X\_train:  
[[ -0.66655417 -0.91305605 -0.79604449 -0.76293154 -0.80471551 -0.83614326  
-1.01496276]  
[ 0.56661206 0.26706335 0.39303214 0.80932131 0.05653262 -0.47287452  
-0.6514148]  
[ 1.08043133 0.66043648 0.80859179 0.66408388 1.06516011 0.82975581  
0.07568112]  
[-1.488665 -1.43755357 -1.30623654 -1.34849198 -1.18698072 -1.25445271  
-0.6514148]  
[ 1.49148674 1.44718275 1.88554961 1.60927987 2.20735015 2.02230471  
0.80277704]]

Head of X\_test:  
[[ 0.15555665 0.39818773 0.16776589 -0.3871585 0.56314917 0.69398871  
1.166325]  
[-0.10135298 -0.51968292 -0.45248775 -0.46323525 -0.35797182 -0.34444617  
-0.6514148]  
[ 0.72075784 0.39818773 0.86310828 0.76321419 1.76981767 0.56556037  
0.43922908]  
[-2.51630352 -2.22429983 -1.60762015 -1.56750081 -1.56464033 -1.61772145  
-1.74205868]  
[ 0.92628555 0.92268524 1.38770091 1.43868352 1.78363449 1.00588611  
0.80277704]]

Menampilkan beberapa baris pertama dari X\_train dan X\_test  
Setelah dilakukan Feature Scaling

# Metode yang Digunakan

```
✓ Metode Linear Regression  
[ ] model = LinearRegression()
```

Metode yang digunakan adalah Regresi linear karena dari dataset yang digunakan dominan cocok menggunakan metode machine learning regresi linear

```
✓ Train The Model  
[ ] model.fit(X_train, Y_train)  
↳ ▾ LinearRegression  
LinearRegression()
```

Melatih Model Regresi Linear dengan melatih model regresi linear menggunakan set pelatihan.

```
✓ Membuat Pengujian  
[ ] Y_pred = model.predict(X_test)  
  
✓ Evaluasi model  
[ ] mse = mean_squared_error(Y_test, Y_pred)  
mae = mean_absolute_error(Y_test, Y_pred)  
r2 = r2_score(Y_test, Y_pred)
```

Setelah model dilatih, kita dapat membuat prediksi pada set pengujian dan mengevaluasi kinerja model.

# Hasil & Analisis

```
[ ] print(f"\nMean Squared Error: {mse}")
print(f"Mean Absolute Error: {mae}")
print(f"R-squared: {r2}")

# Display model coefficients
print("\nModel Coefficients:")
print(f"Intercept: {model.intercept_}")
print(f"Coefficients: {model.coef_}")
```

Menampilkan Hasil dari MSE, MAE & R-Squared

```
Mean Squared Error: 0.00032926035127923286
Mean Absolute Error: 0.013489199784547707
R-squared: 0.9776656009404544
```

Secara keseluruhan, nilai-nilai metrik menunjukkan bahwa model regresi linear sangat akurat untuk data Abalone:

- MSE dan MAE yang rendah menunjukkan bahwa kesalahan prediksi model sangat kecil.
- $R^2$  yang tinggi menunjukkan bahwa model sangat baik dalam menjelaskan variabilitas dalam data.

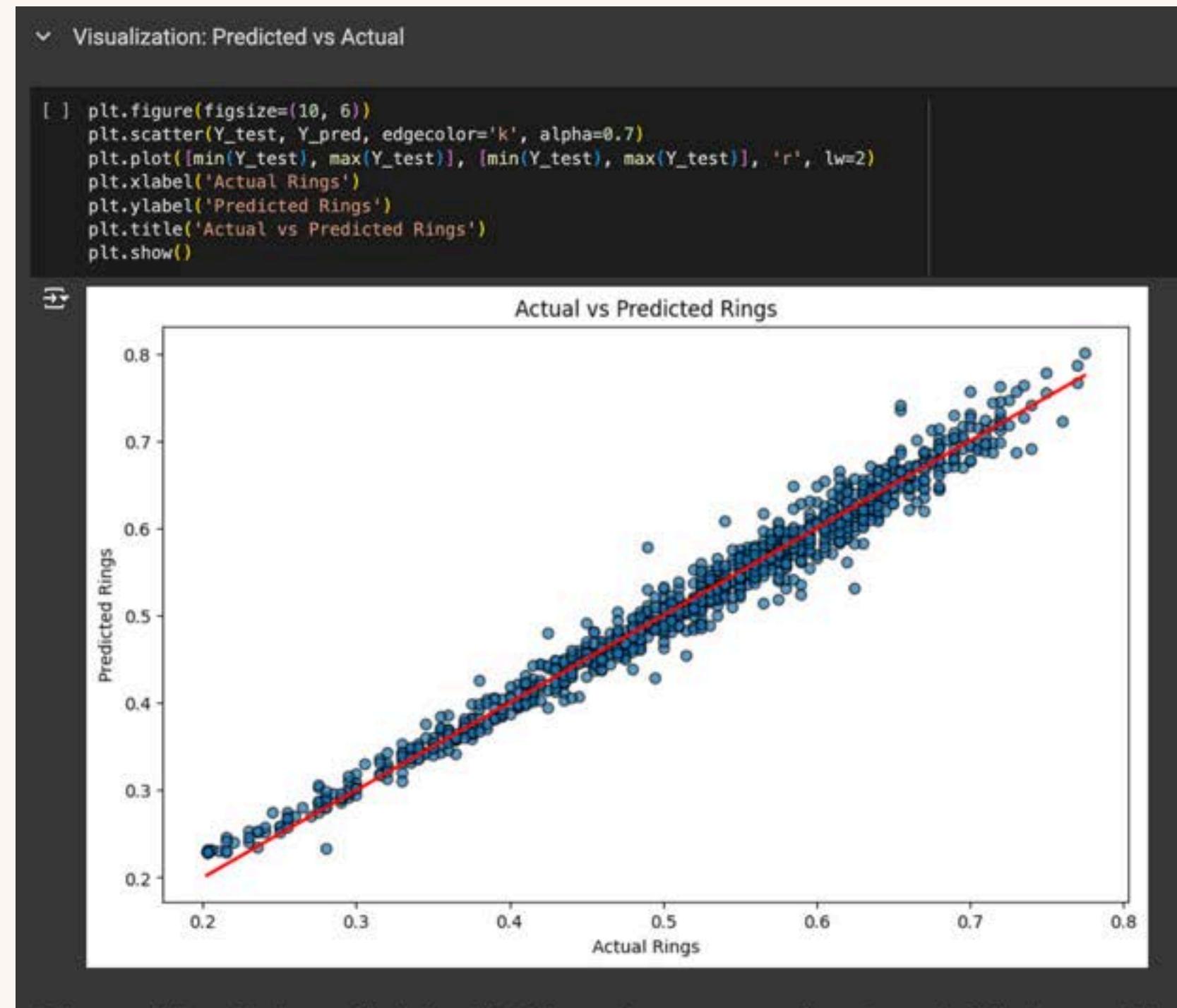
```
Model Coefficients:
Intercept: 0.5263737228607924
Coefficients: [ 1.04230425e-01  2.66359172e-03 -1.54776518e-03  9.62520312e-03
                6.64402589e-03 -3.84436134e-03   6.14202190e-05]
```

Intercept: 0.05965325247164688

- Ini adalah nilai prediksi ketika semua fitur bernilai nol.

Coefficients:

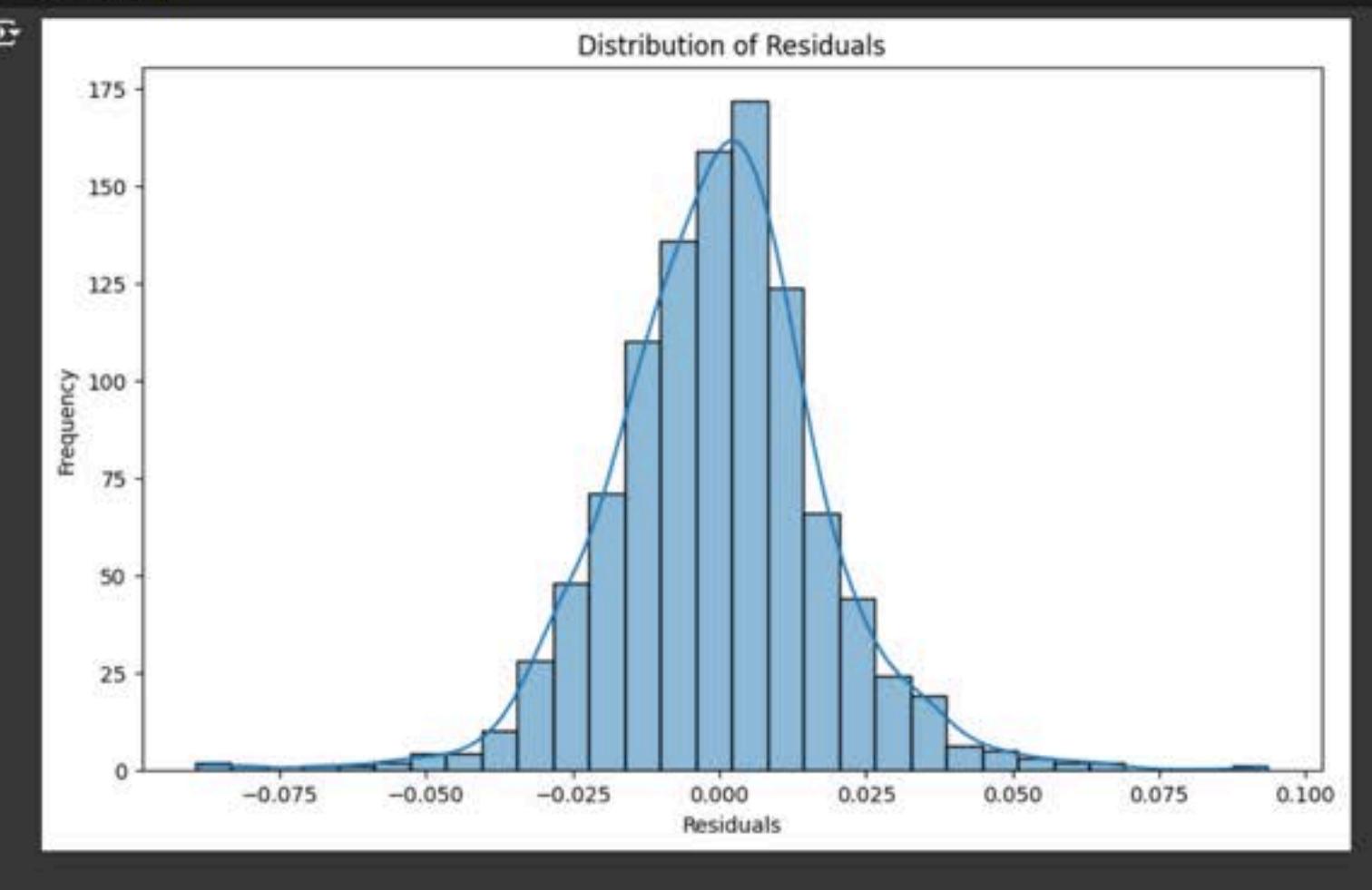
- Setiap nilai koefisien mewakili perubahan dalam target untuk setiap unit perubahan dalam fitur yang sesuai, dengan asumsi fitur lain tetap konstan



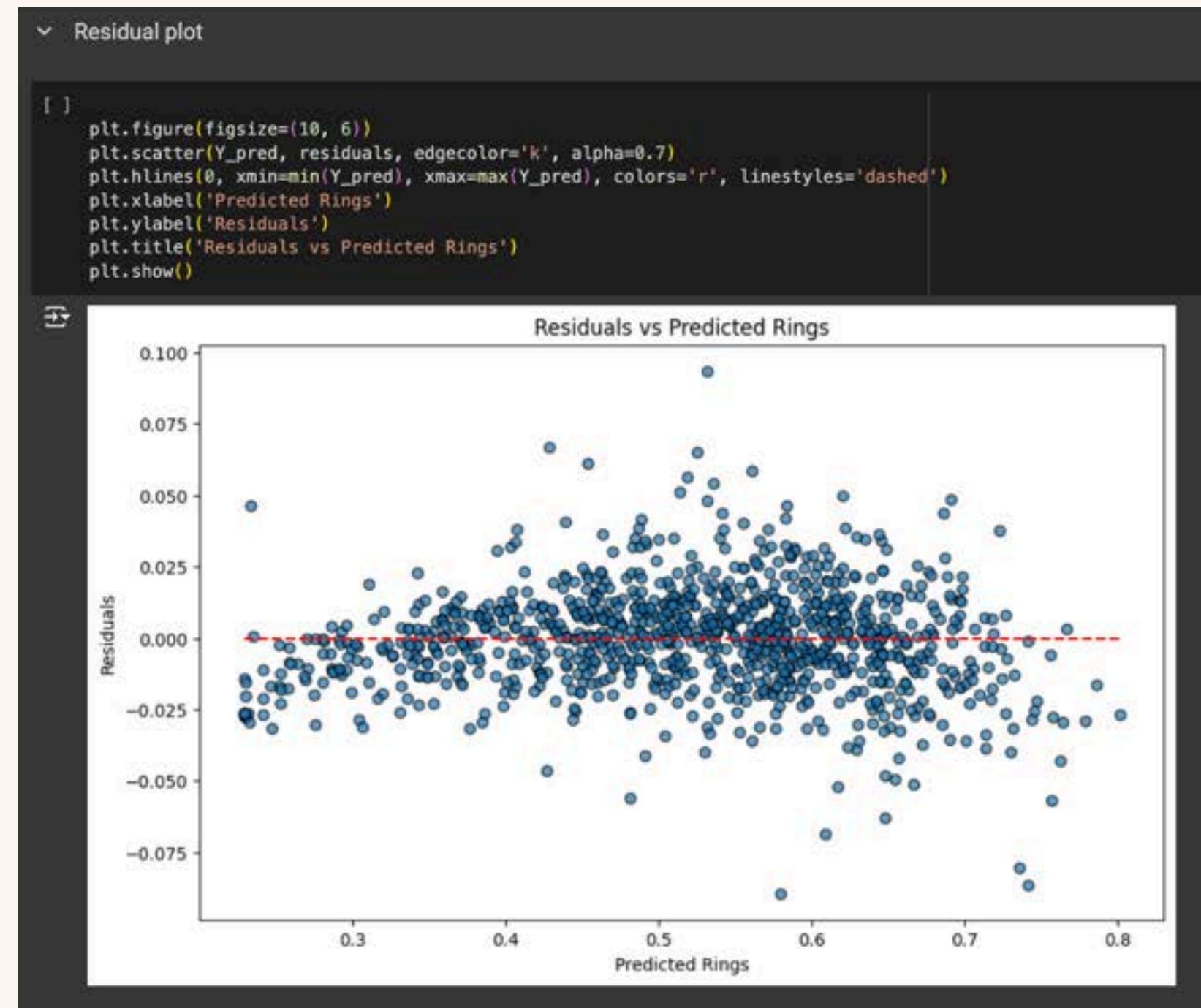
Memvisualisasikan Predicted vs Actual Visualisasi dimana ini menampilkan hubungan antara nilai yang diprediksi oleh model dan nilai sebenarnya (actual values), membantu dalam mengevaluasi kinerja model regresi

```
▼ Visualization: Residuals
```

```
[ ] residuals = Y_test - Y_pred
plt.figure(figsize=(10, 6))
sns.histplot(residuals, kde=True, bins=30)
plt.xlabel('Residuals')
plt.ylabel('Frequency')
plt.title('Distribution of Residuals')
plt.show()
```



Visualisasi distribusi residuals dalam regresi linear digunakan untuk mengevaluasi beberapa aspek penting dari model regresi



Visualisasi residuals vs predicted values dalam regresi linear berperan penting untuk mengevaluasi kualitas dan asumsi model.

## KESIMPULAN

Kesimpulannya, metode machine learning regression dapat memberikan wawasan yang berharga dalam analisis klasifikasi data Abalone, memungkinkan untuk memprediksi umur Abalone berdasarkan fitur-fitur tertentu dan memahami faktor-faktor apa yang mempengaruhi umur mereka. Dengan pemahaman ini, langkah-langkah dapat diambil untuk melindungi dan mengelola populasi Abalone dengan lebih efektif.

Presented by

Muhammad Fakhrul Hizrian &  
Muhammad Rifki Hidayatullah

# Thank you very much!

