Perbandingan Algoritma Regresi Linear dan Random Forest pada Dataset Rating Game

Rama Faisal Muntaha / A11.2022.14082 / A11.4413

1. Ringkasan dan Permasalahan Projek

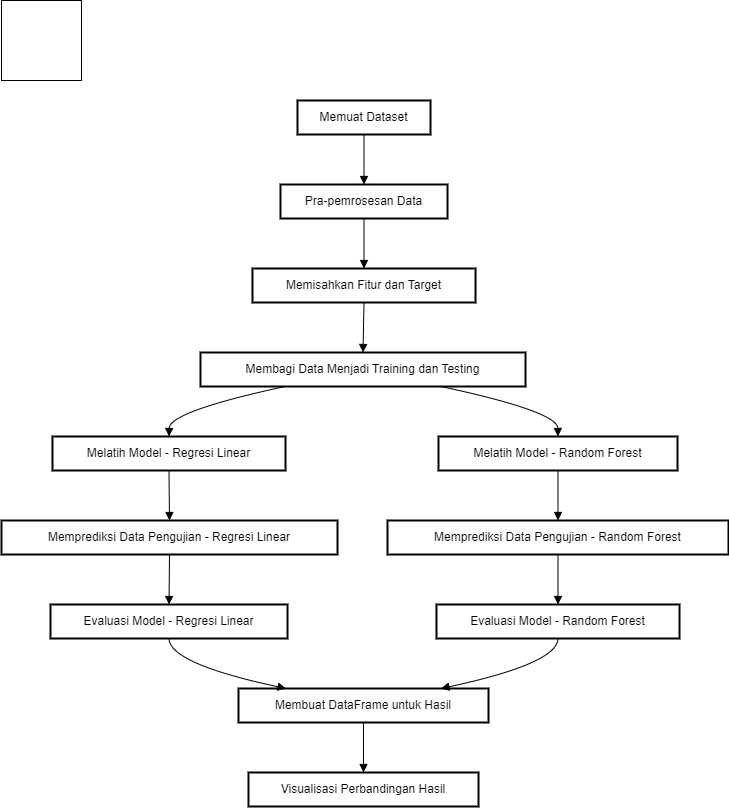
Proyek ini bertujuan untuk membandingkan performa dua algoritma pembelajaran mesin, yaitu Regresi Linear dan Random Forest, dalam memprediksi rating game. Dataset yang digunakan adalah ratings.csv dari Kaggle, yang berisi informasi tentang nama game, developer, genre, dan rating.

Permasalahan: Bagaimana performa model Regresi Linear dan Random Forest dalam memprediksi rating game berdasarkan fitur-fitur yang ada (developer, genre, dll.)?

1. Tujuan yang Ingin Dicapai

* Memahami bagaimana masing-masing model bekerja pada dataset rating game.
* Membandingkan performa model berdasarkan metrik Mean Squared Error (MSE) dan R2 Score.
* Menentukan model yang lebih baik untuk memprediksi rating game.

1. Model / Alur Penyelesaian



1. Penjelasan Dataset, EDA, dan Proses Features Dataset

Dataset yang saya gunakan berasal dari Kaggle.com dengan nama dataset ratings.csv. Dataset ini berisi informasi mengenai rating game pada platform mobile, mencakup

* Game Name: Nama dari game.
* Developer: Nama pengembang game.
* Genre: Kategori atau jenis game.
* Rating: Rating dari game (angka desimal).
* EDA atau Exploratory Data Analysis

1. Memuat Dataset dan menampilkan informasi umum

import pandas as pd

# Memuat dataset

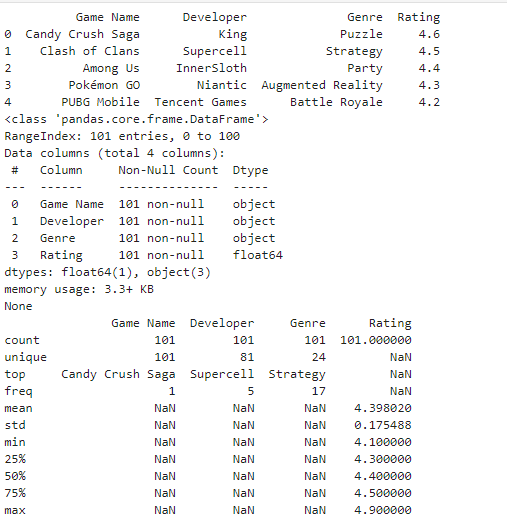
file\_path = '/mnt/data/ratings.csv'

df = pd.read\_csv(file\_path)

print(df.head())

print(df.info())

print(df.describe(include='all'))



1. Analisis Distribusi Rating

import matplotlib.pyplot as plt

import seaborn as sns

# Visualisasi distribusi rating

plt.figure(figsize=(8, 6))

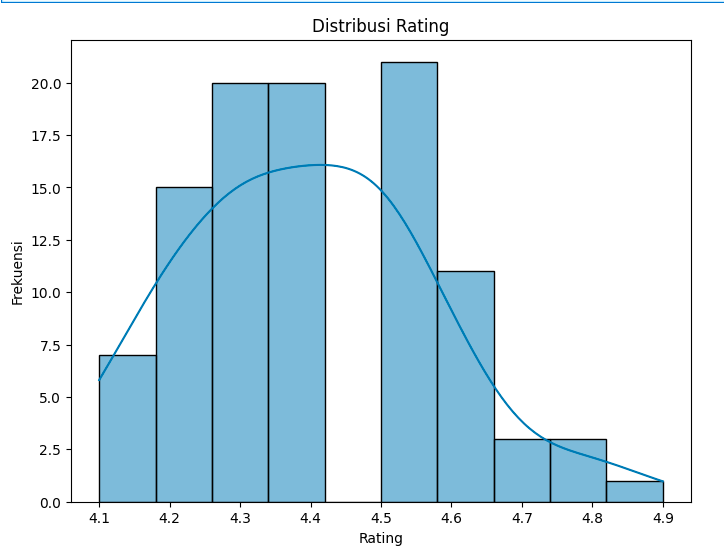
sns.histplot(df['Rating'], bins=10, kde=True)

plt.title('Distribusi Rating')

plt.xlabel('Rating')

plt.ylabel('Frekuensi')

plt.show()



1. Analisis Distribusi Genre

plt.figure(figsize=(10, 6))

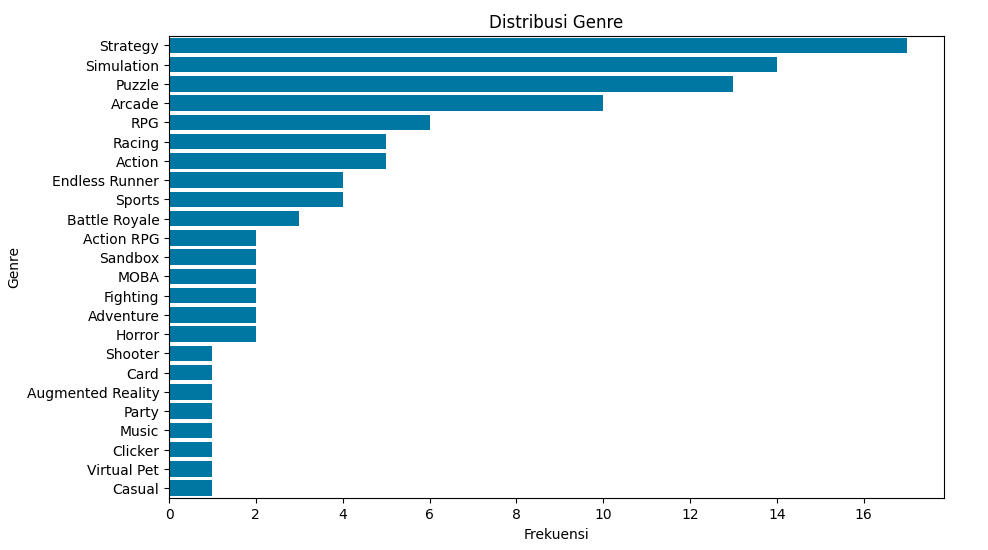
sns.countplot(y=df['Genre'], order=df['Genre'].value\_counts().index)

plt.title('Distribusi Genre')

plt.xlabel('Frekuensi')

plt.ylabel('Genre')

plt.show()



1. Analisis Distribusi Developer

plt.figure(figsize=(10, 6))

top\_developers = df['Developer'].value\_counts().nlargest(10)

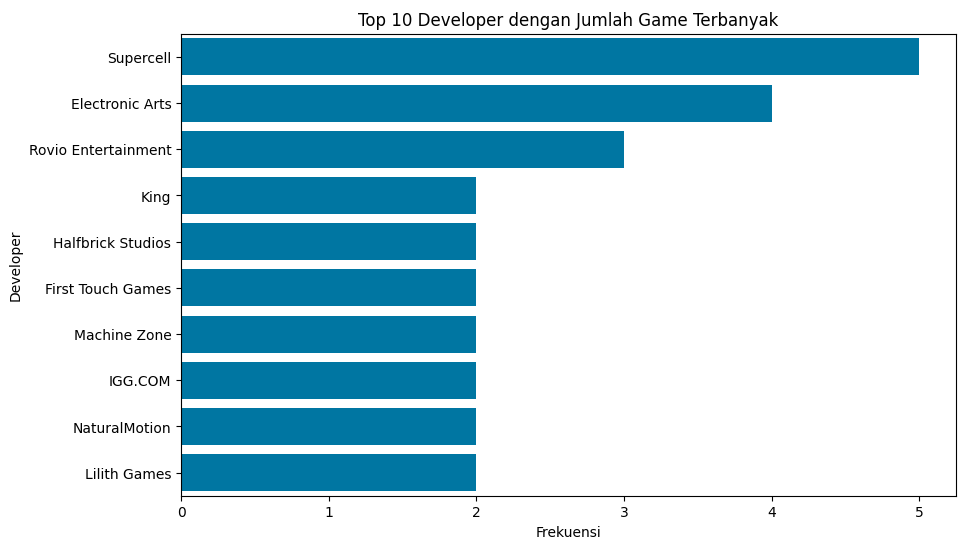
sns.countplot(y=df['Developer'], order=top\_developers.index)

plt.title('Top 10 Developer dengan Jumlah Game Terbanyak')

plt.xlabel('Frekuensi')

plt.ylabel('Developer')

plt.show()



* Proses Features Data

1. Menghapus Missing Value dan Mengonversi Kategori

# Menghapus baris dengan missing values

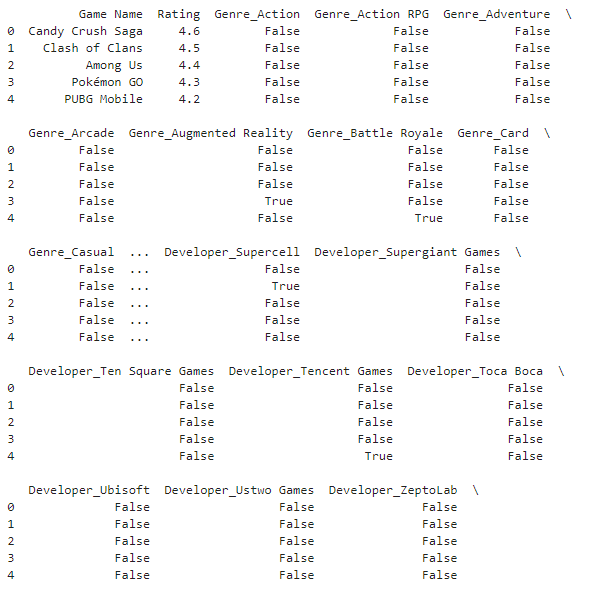
df.dropna(inplace=True)

# Mengonversi kolom kategori menjadi tipe data numerik menggunakan One-Hot Encoding

df = pd.get\_dummies(df, columns=['Genre', 'Developer'])

# Menampilkan beberapa baris pertama setelah pra-pemrosesan

print(df.head())



1. Memisahkan Fitur dan Target

X = df.drop(columns=['Rating', 'Game Name'])

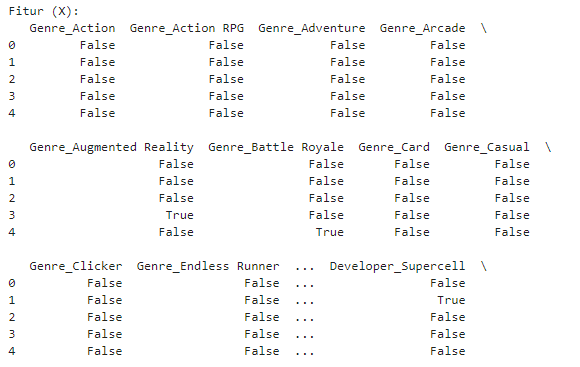
y = df['Rating']

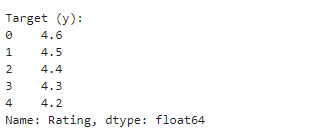
print("Fitur (X):")

print(X.head())

print("\nTarget (y):")

print(y.head())





EDA membantu dalam memahami struktur dan distribusi dataset serta mengidentifikasi anomali atau missing values. Berdasarkan hasil EDA, langkah-langkah dalam Proses Features Data dilakukan untuk mempersiapkan dataset agar siap digunakan dalam model pembelajaran mesin. Sebagai contoh, dalam dataset di atas

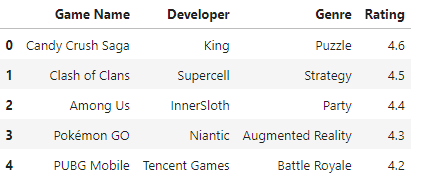
1. Proses Learning / Modeling
2. Proses Algoritma Regresi Linear

* Memuat Dataset

import pandas as pd

df = pd.read\_csv('ratings.csv')

df.head()



* Pra Pemrosesan Data

# Menghapus baris dengan missing values

df.dropna(inplace=True)

print("Jumlah missing values setelah penghapusan:")

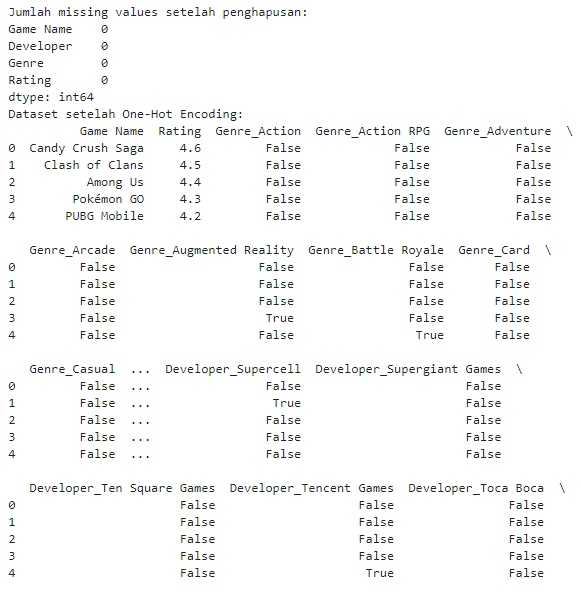
print(df.isnull().sum())

# Mengonversi kolom kategori menjadi tipe data numerik menggunakan One-Hot Encoding

df = pd.get\_dummies(df, columns=['Genre', 'Developer'])

print("Dataset setelah One-Hot Encoding:")

print(df.head())



* Memisahkan Fitur dan Target

# Memisahkan fitur dan target

X = df.drop(columns=['Rating', 'Game Name'])

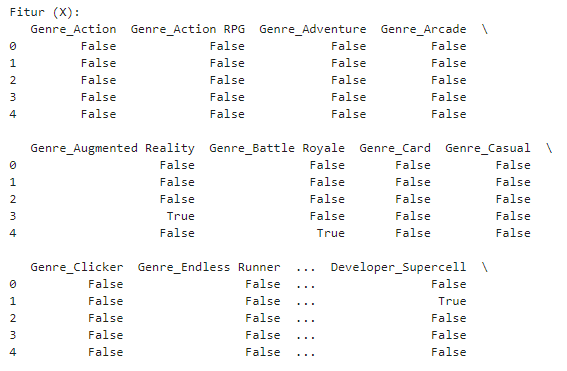
y = df['Rating']

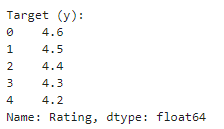
print("Fitur (X):")

print(X.head())

print("\nTarget (y):")

print(y.head())





* Membagi dataset menjadi data pelatihan (training data) dan data pengujian (testing data)

from sklearn.model\_selection import train\_test\_split

# Membagi data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

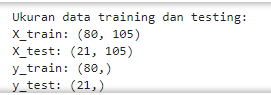
print("Ukuran data training dan testing:")

print(f"X\_train: {X\_train.shape}")

print(f"X\_test: {X\_test.shape}")

print(f"y\_train: {y\_train.shape}")

print(f"y\_test: {y\_test.shape}")



* Melatih model Regresi Linear

from sklearn.linear\_model import LinearRegression

# Melatih model Regresi Linear

lr\_model = LinearRegression()

lr\_model.fit(X\_train, y\_train)

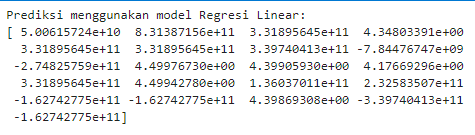
* Prediksi menggunakan model Regresi Linear yang telah dilatih dan menampilkan hasil prediksi

# Memprediksi

y\_pred\_lr = lr\_model.predict(X\_test)

print("Prediksi menggunakan model Regresi Linear:")

print(y\_pred\_lr)



* Evaluasi model Regresi Linear yang telah dilatih dengan menghitung Mean Squad Error (MSE) dan R-Squad (R2)

mse\_lr = mean\_squared\_error(y\_test, y\_pred\_lr)

r2\_lr = r2\_score(y\_test, y\_pred\_lr)

print(f"Regresi Linear - MSE: {mse\_lr}, R2: {r2\_lr}")



* Visualisasi seberapa baik model memprediksi nilai-nilai aktual

plt.figure(figsize=(10, 5))

plt.scatter(y\_test, y\_pred\_lr)

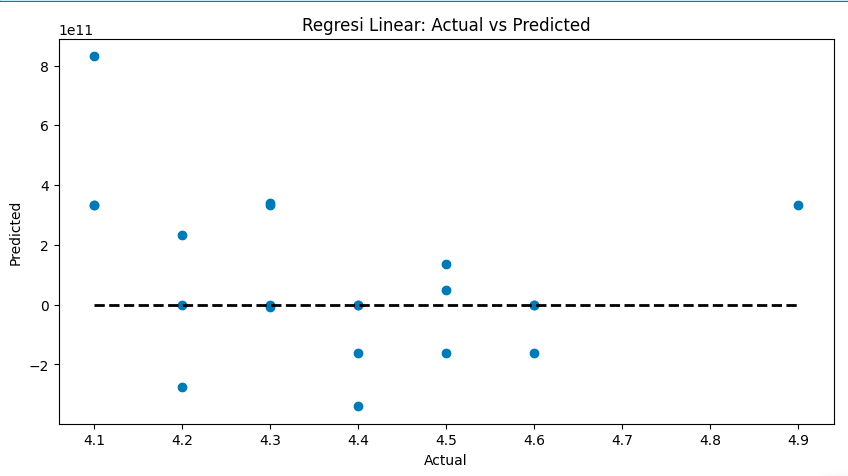
plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'k--', lw=2)

plt.xlabel('Actual')

plt.ylabel('Predicted')

plt.title('Regresi Linear: Actual vs Predicted')

plt.show()



1. Algoritma Random Forest

* Memuat database

import pandas as pd

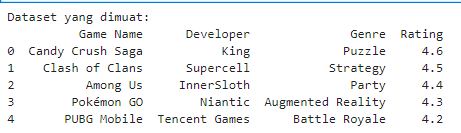
# Memuat dataset

df = pd.read\_csv('ratings.csv')

# Menampilkan beberapa baris pertama

print("Dataset yang dimuat:")

print(df.head())



* Membersihkan dan Memproses data sebelum digunakan model

# Menghapus baris dengan missing values

df.dropna(inplace=True)

print("Jumlah missing values setelah penghapusan:")

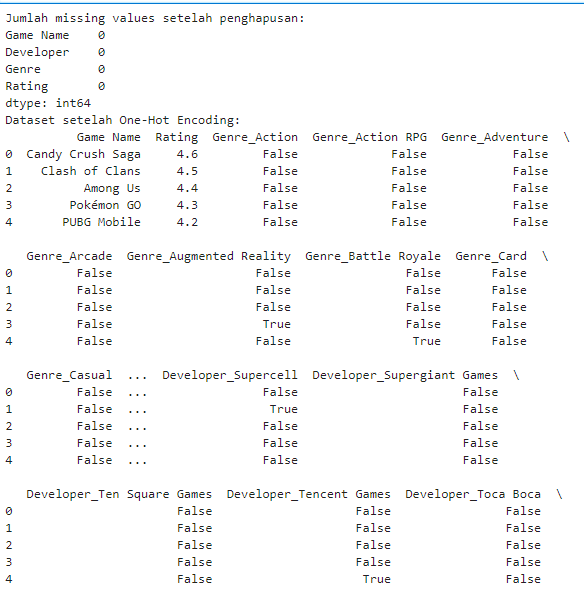
print(df.isnull().sum())

# Mengonversi kolom kategori menjadi tipe data numerik menggunakan One-Hot Encoding

df = pd.get\_dummies(df, columns=['Genre', 'Developer'])

print("Dataset setelah One-Hot Encoding:")

print(df.head())



* Memisahkan fitur (x) dan target (y) dari dataset

X = df.drop(columns=['Rating', 'Game Name'])

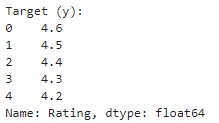
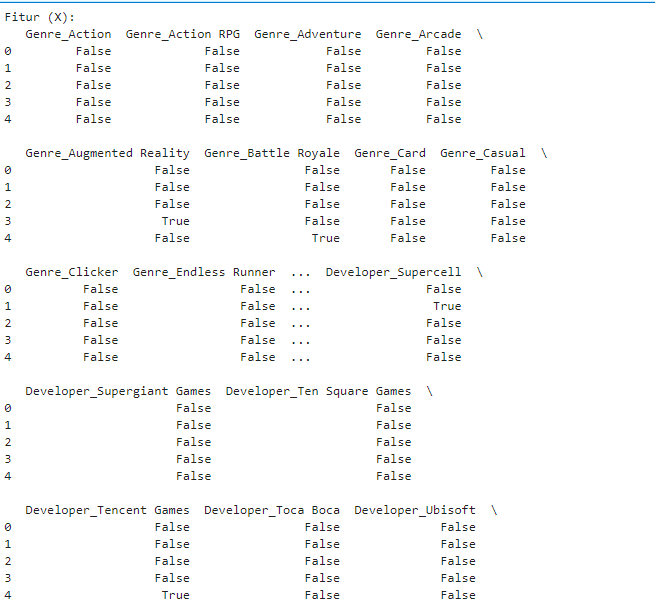
y = df['Rating']

print("Fitur (X):")

print(X.head())

print("\nTarget (y):")

print(y.head())



* Membagi dataset menjadi data training dan data testing

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

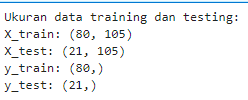
print("Ukuran data training dan testing:")

print(f"X\_train: {X\_train.shape}")

print(f"X\_test: {X\_test.shape}")

print(f"y\_train: {y\_train.shape}")

print(f"y\_test: {y\_test.shape}")



* Melatih model Random Forest Regressor dengan data training yang telah dibagi sebelumnya

from sklearn.ensemble import RandomForestRegressor

rf\_model = RandomForestRegressor(n\_estimators=100, random\_state=42)

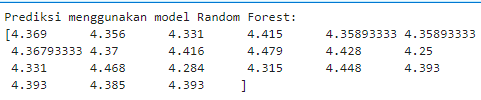
rf\_model.fit(X\_train, y\_train)

* Prediksi model Random Forest yang telah dilatih menggunakan data testing, kemudian menampilkan hasil prediksinya

y\_pred\_rf = rf\_model.predict(X\_test)

print("Prediksi menggunakan model Random Forest:")

print(y\_pred\_rf)



* Evaluasi model random forest

from sklearn.metrics import mean\_squared\_error, r2\_score

mse\_rf = mean\_squared\_error(y\_test, y\_pred\_rf)

r2\_rf = r2\_score(y\_test, y\_pred\_rf)

print(f"Random Forest - MSE: {mse\_rf}, R2: {r2\_rf}")



* Visualisasi hasil prediksi model random forest

import matplotlib.pyplot as plt

plt.figure(figsize=(10, 5))

plt.scatter(y\_test, y\_pred\_rf)

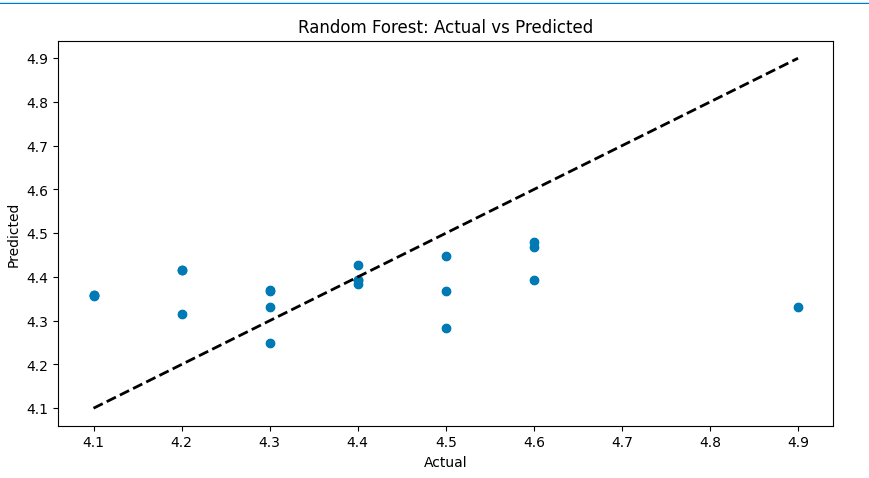
plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'k--', lw=2)

plt.xlabel('Actual')

plt.ylabel('Predicted')

plt.title('Random Forest: Actual vs Predicted')

plt.show()



1. Membandingkan hasil evaluasi dari model Regresi Linear dan Random Forest

Setelah didapatkan model evaluasi dari 2 algoritma diatas kita bisa membandingkan kinerja kedua model secara langsung sehingga kita dapat lebih mudah memilih mana model yang lebih baik berdasarkan metrik yang diberikan.

* Membuat DataFrame yang menyimpan hasil evaluasi dari model Regresi Linear dan Random Forest

results = pd.DataFrame({

'Model': ['Linear Regression', 'Random Forest'],

'MSE': [mse\_lr, mse\_rf],

'R2': [r2\_lr, r2\_rf]

})

print(results)



* Visualisasi perbandingan hasil kedua model

import matplotlib.pyplot as plt

import seaborn as sns

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)

sns.barplot(x='Model', y='MSE', data=results)

plt.title('Mean Squared Error (MSE)')

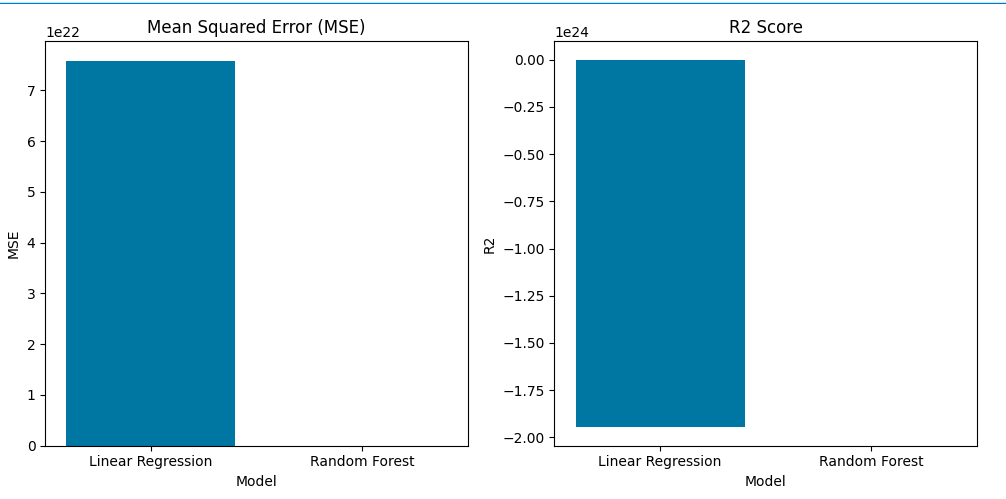
plt.subplot(1, 2, 2)

sns.barplot(x='Model', y='R2', data=results)

plt.title('R2 Score')

plt.tight\_layout()

plt.show()



1. Performa Model

Dari praktikum tersebut kedua model memiliki performa yang berbeda. Performa dari **Model Regresi Linear** memiliki hasil **Man Squad Error 7.584877e+22** dan **R2 score -1.946991e+24**, dan Performa **Model Random Forest** memiliki hasil **Man Squad Error 3.738161e-02** dan **R2 score 4.043705e-02**. Kenapa angka hasil dari Random Forest berbeda dari hasil sebelum dan sesudah dibandingkan? Karena cara penyajian angkanya berbeda, tetapi nilai sebenarnya tetap sama. Sebelum dibandingkan menggunakan format desimal biasa sedangkan setelah dibandingkan menggunakan format ilmiah.

1. Hasil dan Kesimpulan

Berdasarkan hasil evaluasi model yang dilakukan didapat bahwa model **Random Forest** lebih baik dibandingkan model Regresi Linear dalam memprediksi data Rating Game ini. Namun, perlu diingat meskipun model Random Forest ini memiliki MSE yang lebih rendah dan R2 yang sedikit positif, performa secara keseluruhan masih rendah dan mungkin memerlukan optimasi lebih lanjut atau pemilihan fitur yang lebih baik untuk meningkatkan akurasi prediksi.