UTS

Nama : muhammad makhlufi makbullah

Kelas : TK 45 01

NIM : 1103210171

1. Import Libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.neighbors import KNeighborsClassifier, KNeighborsRegressor

from sklearn.metrics import accuracy_score, mean_squared_error, confusion_matrix, ConfusionMatrixDisplay

- pandas: Untuk memproses data tabular (CSV file).
- numpy: Untuk perhitungan matematika seperti menghitung tren linier.
- matplotlib: Untuk membuat visualisasi data (scatter plot, trend line).
- scikit-learn: Untuk membangun model pembelajaran mesin seperti K-Nearest Neighbors (KNN) dan menghitung metrik evaluasi.

2. Unggah dan Baca Dataset

from google.colab import files

uploaded = files.upload() # Anda akan diminta untuk mengunggah file

data = pd.read_csv('Salary_Data.csv')

print(data.head())

print(data.dtypes)

- files.upload: Fungsi ini digunakan untuk mengunggah file di Google Colab.
- pd.read_csv: Membaca file CSV dan menyimpannya dalam bentuk DataFrame.
- data.head(): Menampilkan 5 baris pertama dari dataset.
- data.dtypes: Mengecek tipe data setiap kolom dalam DataFrame.

3. Visualisasi Data

Scatter Plot

plt.scatter(data['YearsExperience'], data['Salary'], color='blue')

plt.title('Years of Experience vs Salary')

plt.xlabel('Years of Experience')

plt.ylabel('Salary')

plt.grid(True)

plt.show()

- Membuat plot untuk melihat hubungan antara YearsExperience (pengalaman kerja) dan Salary (gaji).

Trend Line

m, b = np.polyfit(data['YearsExperience'], data['Salary'], 1)

plt.scatter(data['YearsExperience'], data['Salary'], color='blue')

*plt.plot(data['YearsExperience'], m*data['YearsExperience'] + b, color='red') # Garis tren*

plt.title('Years of Experience vs Salary with Trend Line')

plt.xlabel('Years of Experience')

plt.ylabel('Salary')

plt.grid(True)

plt.show()

- np.polyfit menghitung linear regression sederhana (kemiringan m dan intercept b).
- Menambahkan garis tren linier ke plot sebelumnya.

4. Split Dataset

X = data[['YearsExperience']] # Fitur (independent variable)

y = data['Salary'] # Target (dependent variable)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

X: Berisi kolom YearsExperience (fitur yang digunakan untuk prediksi).

y: Berisi kolom Salary (target yang akan diprediksi).

train_test_split: Membagi dataset menjadi train set (80%) dan test set (20%).

5. K-Nearest Neighbors Regressor

Train Model

knn = KNeighborsRegressor(n_neighbors=3)

knn.fit(X_train, y_train)

- Menggunakan model KNN Regressor dengan 3 tetangga terdekat (n_neighbors=3).

fit melatih model dengan data training.

Predict Test Data

y_pred = knn.predict(X_test)

- predict menghasilkan prediksi untuk data X_test.

6. Error Metrics

Root Mean Squared Error

mse = mean_squared_error(y_test, y_pred)

rmse = np.sqrt(mse)

print("Root Mean Squared Error:", rmse)

- mean_squared_error menghitung rata-rata kuadrat error (MSE).
- np.sqrt mengambil akar kuadrat dari MSE untuk mendapatkan Root Mean Squared Error (RMSE).

7. Kesalahan di Bagian Confusion Matrix

Bagian ini ada kesalahan dalam penggunaan KNN Regressor:

cm = confusion_matrix(y_test, y_pred)

disp = ConfusionMatrixDisplay(confusion_matrix=cm)

disp.plot(cmap='Blues')

8. Kolom dengan Tipe Float

float_columns = data.select_dtypes(include=['float64']).columns

print("Kolom bertipe float64:", float_columns.tolist())

- Mengecek kolom dalam DataFrame yang memiliki tipe data float64.

9. Ringkasan Statistik dengan Gradien Warna

data.describe().loc[['min','50%','mean','max','std']].T.style.background_gradient(axis=1)

- **data.describe()**: Memberikan ringkasan statistik untuk setiap kolom numerik dalam dataset, termasuk nilai **min**, **mean**, **50% (median)**, **max**, dan **std** (deviasi standar).
- **.loc[['min','50%','mean','max','std']]**: Memilih baris tertentu dari ringkasan statistik untuk ditampilkan.
- **.T**: Transpose DataFrame agar baris menjadi kolom.
- **.style.background_gradient(axis=1)**: Menambahkan gradien warna berdasarkan nilai untuk setiap kolom, membuat perbedaan nilai lebih terlihat.

10. Cek Nilai NULL (Data Hilang)

missing_values = data.isnull().sum()

missing_values

- **data.isnull()**: Mengembalikan DataFrame boolean yang menunjukkan apakah sebuah nilai adalah **null** atau tidak.
- **.sum()**: Menjumlahkan nilai **True** (null) di setiap kolom untuk mengetahui jumlah data yang hilang.

11. Heatmap Korelasi Antar Fitur

plt.figure(figsize=(16, 12))

correlation_matrix = data.corr() # Menghitung matriks korelasi

sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap='coolwarm', linewidths=0.5)

plt.title('Heatmap Korelasi Antar Fitur', fontsize=16)

plt.xticks(fontsize=12, rotation=45)

plt.yticks(fontsize=12)

plt.show()

a. Membuat Matriks Korelasi

- **data.corr()**: Menghitung korelasi antara kolom numerik dalam dataset. Korelasi adalah nilai antara -1 hingga 1:
    - **1**: Hubungan positif sempurna.
    - **-1**: Hubungan negatif sempurna.
    - **0**: Tidak ada hubungan.

b. Visualisasi Heatmap

- **sns.heatmap()**: Membuat visualisasi matriks korelasi dalam bentuk heatmap.
    - **annot=True**: Menampilkan nilai korelasi di setiap kotak.
    - **fmt=".2f"**: Menampilkan nilai korelasi dengan dua angka desimal.
    - **cmap='coolwarm'**: Menggunakan palet warna biru (negatif) ke merah (positif).
    - **linewidths=0.5**: Memberi garis pemisah antar kotak untuk estetika.

c. Pengaturan Visualisasi

- **plt.figure(figsize=(16, 12))**: Menentukan ukuran gambar.
- **plt.title()**: Memberi judul heatmap.
- **plt.xticks() dan plt.yticks()**: Mengatur ukuran dan rotasi label di sumbu x dan y.