PESAN KEVIN LATIHAN GANTENG 👍

Latihan Soal Machine Learning Data Preprocessing :

1. Silahkan gabungkan semua dataset yang telah diberikan, lakukan normalisasi, data cleaning dan data preprocessing
2. Lakukan feature engineering jika diperlukan
3. Print head, tail dan jumlah data beserta kolom-kolomnya setelah ke2 langkah di atas
4. Hitung Head-to-Head ketika Arya melawan Dennis, berapa kali Arya menang dan berapa kali Dennis Menang? Jika ada match mereka ber2 berpasangan maka tidak dihitung
5. Hitung berapa kali David menang dengan skor 2-0?
6. Hitung berapa kali Moel kalah dengan skor 1-2?

Data Training and Testing :

1. Split data untuk training dan testing, persentasenya silahkan ditentukan sendiri
2. Lakukan Training dan testing data untuk :

- Memprediksi jumlah skor jika Arya/Dennis melawan David/Moel jika mereka bertanding sebanyak 10 match? Asumsikan bahwa ke-4 pemain tersebut sehat wal afiat selama 10 match

- Jika Arya sedang Sakit dan dipasangkan dengan Moel yang sedang pemulihan, lalu mereka akan bertanding secara random sebanyak 5 match melawan ke6 pemain lainnya yang dipasangkan secara acak, lakukan prediksi skornya, asumsikan ke6 pemain lainnya sehat.

- Dari 8 pemain yang tersedia, pelatih ingin mengirimkan 3 pasangan terbaik yang bisa dikirimkan untuk bertanding, silahkan cari 3 pasangan tersebut, untuk studi kasus kali ini silahkan kesampingkan kondisi pemain.

Note : Disarankan untuk mencoba semua algoritma yang dapat digunakan untuk memprediksi masing-masing study case

Evaluation :
Hitung dan gunakan metrics yang sesuai untuk menguji kualitas model machine learning yang telah ditraining berdasarkan masing-masing studi kasus.

1. Data Preprocessing

a. Combining Datasets & Cleaning

What you need to do:

- **Merge datasets:** Suppose you have several CSV files. You can use Python's `pandas.concat()` to combine them into one DataFrame.
- **Normalization:** This means scaling numerical values so that they're on a similar scale. For example, you might use MinMaxScaler from scikit-learn.
- **Data Cleaning:** Remove or fill in missing values and remove any outliers or irrelevant data. For instance, using `DataFrame.dropna()` or `DataFrame.fillna()`.

Example:

python
CopyEdit

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# Merge datasets (assuming data1, data2, etc. are DataFrames)
data = pd.concat([data1, data2, data3], ignore_index=True)

# Clean data (drop rows with missing values)
data = data.dropna()

# Normalize numerical columns
scaler = MinMaxScaler()
data[['col1', 'col2']] = scaler.fit_transform(data[['col1', 'col2']])
```

b. Feature Engineering

Sometimes your raw data doesn't tell the whole story. You might need to create new features (columns) that help your model. For example, if you have a date column, you might extract the month or day.

Example:

python

CopyEdit
```
data['month'] = pd.to_datetime(data['date']).dt.month
```

c. Printing Data Overview

You'll want to see the first few rows (head), last few rows (tail), and a summary of your DataFrame.

- **Head:** `data.head()`
- **Tail:** `data.tail()`
- **Count of data and columns:** `data.info()` or `data.shape`

d. Specific Calculations

- **Head-to-Head for Arya vs. Dennis:**
  Count matches where only one of them played solo (not when paired together). You can filter your DataFrame by conditions (e.g., `data[(data['player1']=='Arya') & (data['player2']=='Dennis')]`) and then count wins based on the score column.
- **David wins with score 2-0:**
  Filter rows where the winner is David and the score is exactly 2-0.
- **Moel loses with score 1-2:**
  Filter rows where Moel is the loser and the score is exactly 1-2.

Example pseudocode:

python
CopyEdit
```
# Count Arya wins (only when not paired together)
arya_wins = data[(data['player1'] == 'Arya') & (data['player2'] ==
'Dennis') & (data['score'] indicates Arya win)].shape[0]
dennis_wins = data[(data['player1'] == 'Arya') & (data['player2'] ==
'Dennis') & (data['score'] indicates Dennis win)].shape[0]

# Count David wins with score 2-0
```

```
david_2_0_wins = data[(data['winner'] == 'David') & (data['score'] ==
'2-0')].shape[0]

# Count Moel losses with score 1-2
moel_losses_1_2 = data[(data['loser'] == 'Moel') & (data['score'] ==
'1-2')].shape[0]
```

Note: The actual filtering depends on how your data is structured (columns names, score formats, etc.).

2. Data Training and Testing

a. Splitting Data

Use scikit-learn's `train_test_split()` to divide your data into training and testing sets. You can choose any split (e.g., 70% training and 30% testing).

Example:

python
CopyEdit
```
from sklearn.model_selection import train_test_split

X = data.drop('target', axis=1)  # features
y = data['target']  # what you want to predict (e.g., score)

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)
```

b. Training Models for Different Study Cases

You have three different scenarios:

1. **Predicting scores for Arya/Dennis vs. David/Moel in 10 matches:**
   - Here, you may use regression models (like Linear Regression) to predict scores.
2. **Scenario with Arya (sick) & Moel (in recovery) paired against 6 other players randomly in 5 matches:**

  - Again, you could build a regression model but with new features indicating player conditions (e.g., healthy vs. sick). You might simulate randomness if needed.

3. **Choosing 3 best pairs from 8 players:**

    - This could be approached as an optimization problem. You might score each possible pair (based on historical win rates or performance metrics) and choose the top 3.

    - Alternatively, you could use clustering or ranking algorithms to decide the best pairs.

For each scenario, remember:

- Train your model using the training set.

- Test and evaluate the model on the testing set.

c. Evaluation Metrics

Depending on your prediction task (most likely regression), common evaluation metrics include:

- **Mean Squared Error (MSE):** Measures the average squared difference between predicted and actual values.

- **Mean Absolute Error (MAE):** Measures the average absolute difference.

- **R-squared (R^2):** Indicates the proportion of variance explained by the model.

Example:

python
CopyEdit
```
from sklearn.metrics import mean_squared_error, mean_absolute_error,
r2_score

y_pred = model.predict(X_test)
print("MSE:", mean_squared_error(y_test, y_pred))
print("MAE:", mean_absolute_error(y_test, y_pred))
print("R^2:", r2_score(y_test, y_pred))
```

Extra Exercise Questions for You

Question 1: Data Cleaning

Q: Why is it important to handle missing data before training your machine learning model?
A: Handling missing data prevents the model from learning from incomplete information, reduces bias, and improves the accuracy of predictions. It also avoids errors during model training since many algorithms cannot handle missing values.

Question 2: Feature Engineering

Q: What is feature engineering and how can it improve model performance?
A: Feature engineering involves creating new input features or transforming existing ones to better represent the underlying problem. It can improve model performance by highlighting patterns and relationships that the raw data might not clearly show.

Question 3: Model Evaluation

Q: What are the differences between MSE and MAE, and when might you prefer one over the other?
A: MSE penalizes larger errors more than MAE due to the squaring of differences, making it more sensitive to outliers. MAE provides a more straightforward average error. You might choose MAE if you want a more robust metric in the presence of outliers, whereas MSE can be useful if large errors are particularly undesirable.

Question 4: Splitting Data

Q: What is the purpose of splitting your dataset into training and testing sets?
A: Splitting data helps to evaluate how well your model generalizes to unseen data. The training set is used to build the model, while the testing set provides an unbiased evaluation of its performance.

Question 5: Handling Special Cases

Q: In the scenario where Arya is sick and Moel is recovering, how might you adjust your model or data to account for their conditions?
A: You could introduce additional features that represent the players' conditions (e.g., a binary flag or a performance degradation factor). Alternatively, you could simulate the impact of these conditions by altering historical performance data or using domain knowledge to adjust predictions.

Final Thoughts

This exercise is designed to let you practice:

- Combining and cleaning data.
- Creating new features (feature engineering).
- Filtering and computing statistics based on specific conditions.
- Splitting data, training models, and evaluating them using proper metrics.

PUNYA ZIDANE BREU

With data FOR LAB:

Jupyter Notebook: `1-3_Preprocessing_Badminton.ipynb`

python
CopyEdit

```
# ================================
# 📦 STEP 1: Import Library
# ================================
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# ================================
# 📥 STEP 2: Load & Gabung Dataset
# ================================
df1 = pd.read_csv("Badminton_Match_Result_Dataset_1.csv",
delimiter=';')
df2 = pd.read_csv("Badminton_Match_Result_Dataset_2.csv",
delimiter=';')
df3 = pd.read_csv("Badminton_Match_Result_Dataset_3.csv",
delimiter=';')

# Gabungkan ketiga dataset
df = pd.concat([df1, df2, df3], ignore_index=True)

# ================================
# 🧼 STEP 3: Data Cleaning
# ================================
# Drop kolom yang tidak dipakai
df = df.drop(columns=[
    'tim_1', 'tim_2', 'kondisi_team1_player1',
    'kondisi_team1_player2', 'kondisi_team2_player1',
    'kondisi_team2_player2', 'skor'
], errors='ignore')
```

```python
# Drop baris yang tidak punya skor
df = df.dropna(subset=['score'])

# Bersihkan format skor (hapus tanda petik tunggal)
df['score'] = df['score'].str.replace("'", "").str.strip()

# Filter hanya skor yang valid (mengandung tanda '-')
df = df[df['score'].str.contains('-')]

# Pisahkan skor jadi 2 kolom angka
df[['team_1_score', 'team_2_score']] = df['score'].str.split('-',
expand=True).astype(int)

# Drop kolom skor lama
df = df.drop(columns=['score'])

# ================================
# 🔄 STEP 4: Normalisasi (Min-Max Scaling)
# ================================
# Kita bisa normalisasi skor
scaler = MinMaxScaler()
df[['team_1_score_norm', 'team_2_score_norm']] =
scaler.fit_transform(df[['team_1_score', 'team_2_score']])

# ================================
# 🖨️ STEP 5: Print Info
# ================================
print("📌 HEAD:")
print(df.head())

print("\n📌 TAIL:")
print(df.tail())

print(f"\n📊 JUMLAH DATA: {len(df)} baris")
print(f"🧾 KOLOM: {df.columns.tolist()}")
```

Step 1: Gabungkan dan Bersihkan Data

Dataset kamu sebenarnya sudah digabung, **tapi ada masalah format** — semua kolom masih tergabung dalam satu kolom (disebabkan oleh delimiter `;` bukan `,`).

Jadi langkah pertama kita adalah:

1. Pisahkan kolom-kolom menggunakan delimiter `;`
2. Bersihkan data (hapus kolom yang kosong atau duplikat)
3. Normalisasi nama kolom biar konsisten

Let's go, kita bersihkan datanya dulu 🔧.

Datanya sudah berhasil kita bersihkan dan kolom-kolom pentingnya sudah bisa kita olah. Ada beberapa kolom duplikat atau kosong (seperti `tim_1`, `tim_2`, `skor`, dll) — nanti kita buang.

Step 2: Bersihkan Data (Cleaning)

Target:

- Buang kolom yang kosong
- Ubah skor menjadi angka (misalnya `'2-1 → (2, 1)`)

Let's bersihin lebih lanjut 💪

Sepertinya ada beberapa baris skor yang kosong atau salah format, sehingga gagal dikonversi ke integer (`NaN tidak bisa dikonversi ke int`).

Solusi:

Kita bersihkan baris-baris dengan skor kosong atau invalid sebelum split dan konversi.

Yuk lanjut bersihin datanya dulu.

It seems like I can’t do more advanced data analysis right now. Please try again later.

But no worries — kamu masih bisa lanjutin sendiri kok! Ini aku kasih langkah-langkah manual yang bisa kamu salin ke Jupyter Notebook atau Google Colab:

Langkah Manual Membersihkan Skor

python
CopyEdit

```
# Drop rows yang kosong di kolom 'score'
cleaned_df = cleaned_df.dropna(subset=['score'])

# Hapus tanda petik jika ada
cleaned_df['score'] = cleaned_df['score'].str.replace("'", "")

# Ambil hanya yang format skornya valid (mengandung '-')
cleaned_df = cleaned_df[cleaned_df['score'].str.contains('-')]

# Pisahkan skor jadi dua kolom integer
cleaned_df[['team_1_score', 'team_2_score']] =
cleaned_df['score'].str.split('-', expand=True).astype(int)

# Hapus kolom 'score' yang lama
cleaned_df = cleaned_df.drop(columns=['score'])

# Tampilkan hasil
print(cleaned_df.head())
```

Kalau kamu mau lanjut ngerjain:

- Hitung kemenangan Arya vs Dennis
- Hitung berapa kali David menang 2-0
- Hitung berapa kali Moel kalah 1-2
-

nomor 4–6:

Jupyter Notebook: `Machine_Learning_Preprocessing.ipynb`

python
CopyEdit

```
# =================================
# 🧹 Step 1: Import Library
# =================================
import pandas as pd

# =================================
# 📂 Step 2: Load Dataset
# Ganti path jika file ada di lokasi berbeda
# =================================
df1 = pd.read_csv("Badminton_Match_Result_Dataset_1.csv",
delimiter=';')
df2 = pd.read_csv("Badminton_Match_Result_Dataset_2.csv",
delimiter=';')
df3 = pd.read_csv("Badminton_Match_Result_Dataset_3.csv",
delimiter=';')

# Gabungkan semua dataset
df = pd.concat([df1, df2, df3], ignore_index=True)

# =================================
# 🔍 Step 3: Bersihkan Kolom & Format Skor
# =================================
# Buang kolom tak penting
df = df.drop(columns=['tim_1', 'tim_2', 'kondisi_team1_player1',
                      'kondisi_team1_player2',
'kondisi_team2_player1',
                      'kondisi_team2_player2', 'skor'],
errors='ignore')

# Bersihkan kolom skor
df = df.dropna(subset=['score'])  # hapus yang kosong
df['score'] = df['score'].str.replace("'", "")  # hapus tanda petik
df = df[df['score'].str.contains('-')]  # hanya yang valid
```

```
# Pisahkan skor menjadi 2 kolom angka
df[['team_1_score', 'team_2_score']] = df['score'].str.split('-',
expand=True).astype(int)
df = df.drop(columns=['score'])

# Lihat hasil
print("Data setelah preprocessing:")
print(df.head())
print(df.tail())
print("Jumlah Data:", len(df))
print("Kolom:", df.columns.tolist())

# ================================
# ⚔️ Step 4: Arya vs Dennis Head-to-Head
# ================================
# Ambil baris yang melibatkan Arya & Dennis tapi tidak dalam tim yang
sama
head_to_head = df[
    (((df['team1_player1'] == 'Arya') | (df['team1_player2'] ==
'Arya')) &
     ((df['team2_player1'] == 'Dennis') | (df['team2_player2'] ==
'Dennis')))
    |
    (((df['team2_player1'] == 'Arya') | (df['team2_player2'] ==
'Arya')) &
     ((df['team1_player1'] == 'Dennis') | (df['team1_player2'] ==
'Dennis')))
]

# Hitung kemenangan
arya_win = ((head_to_head['team1_player1'] == 'Arya') |
(head_to_head['team1_player2'] == 'Arya')) & \
           (head_to_head['team_1_score'] >
head_to_head['team_2_score'])

dennis_win = ((head_to_head['team1_player1'] == 'Dennis') |
(head_to_head['team1_player2'] == 'Dennis')) & \
             (head_to_head['team_1_score'] >
head_to_head['team_2_score'])

arya_win_count = arya_win.sum()
dennis_win_count = dennis_win.sum()

print(f"\n🔢 Arya vs Dennis:")
print(f"Arya menang: {arya_win_count} kali")
print(f"Dennis menang: {dennis_win_count} kali")

# ================================
# 🏸 Step 5: Berapa Kali David Menang 2-0
# ================================
david_win_2_0 = df[
    (((df['team1_player1'] == 'David') | (df['team1_player2'] ==
'David')) & (df['team_1_score'] == 2) & (df['team_2_score'] == 0)) |
    (((df['team2_player1'] == 'David') | (df['team2_player2'] ==
'David')) & (df['team_2_score'] == 2) & (df['team_1_score'] == 0))
]

print(f"\n🏆 David menang 2-0 sebanyak: {len(david_win_2_0)} kali")

# ================================
# 💥 Step 6: Moel Kalah 1-2
# ================================
moel_lose_1_2 = df[
    (((df['team1_player1'] == 'Moel') | (df['team1_player2'] ==
'Moel')) & (df['team_1_score'] == 1) & (df['team_2_score'] == 2)) |
    (((df['team2_player1'] == 'Moel') | (df['team2_player2'] ==
'Moel')) & (df['team_2_score'] == 1) & (df['team_1_score'] == 2))
]

print(f"\n😓 Moel kalah 1-2 sebanyak: {len(moel_lose_1_2)} kali")
```

📌 Catatan:

- Simpan ketiga file CSV (`Dataset_1.csv`, dst) di folder yang sama dengan `.ipynb`-mu
- Kamu bisa pakai Jupyter Notebook, Google Colab, atau VSCode + Jupyter