FINAL PROJECT MCI

Prediksi Hasil Pertandingan NBA Berdasarkan Statistik Pemain

Data Diri

Ammar Ghozy Tanumijaya

5025231203

Teknik Informatika

Daftar Isi

Latar Belakang

Rumusan Masalah & Tujuan

Dataset & Kode

Preprocessing & EDA

Metodologi

Hasil Evaluasi

Deployment

Tampilan Aplikasi

Kesimpulan & Saran

Referensi

Latar Belakang

Olahraga, khususnya NBA, tidak hanya menjadi hiburan dan ajang kompetisi, tetapi juga industri besar yang menghasilkan data dalam jumlah sangat besar setiap musimnya. Statistik pemain seperti poin, assist, steal, dan blok menjadi indikator penting dalam menilai performa tim. Ratusan pertandingan yang berlangsung sepanjang musim menciptakan peluang besar untuk menganalisis dan memahami pola permainan melalui data.

Seiring berkembangnya teknologi, teknik data mining dan machine learning memungkinkan kita mengolah data tersebut untuk memprediksi hasil pertandingan secara lebih akurat. Dengan belajar dari data historis, model prediktif dapat memberikan informasi berharga yang bermanfaat bagi pelatih, analis, dan bahkan sektor komersial seperti media dan taruhan olahraga. Hal ini mendorong perlunya sistem prediksi berbasis data yang cepat, akurat, dan dapat digunakan secara praktis.

Rumusan Masalah & Tujuan

Banyaknya data statistik dari setiap pertandingan NBA memunculkan pertanyaan penting: apakah hasil pertandingan dapat diprediksi hanya dengan melihat performa pemain dan tim lawan? Penelitian ini bertolak dari pertanyaan tersebut, dengan tujuan menguji kemungkinan membuat sistem prediktif berbasis data yang dapat mengklasifikasikan hasil pertandingan menjadi menang atau kalah. Jika terbukti berhasil, pendekatan ini dapat membuka peluang luas dalam penerapan analitik data di dunia olahraga.

Untuk menjawabnya, penelitian ini membangun model klasifikasi menggunakan dua algoritma machine learning populer, yaitu Decision Tree dan Random Forest. Kedua model dibandingkan untuk melihat mana yang paling akurat dan efisien. Setelah model terbaik ditemukan, sistem prediksi ini kemudian diintegrasikan ke dalam aplikasi web interaktif, sehingga pengguna dapat melakukan prediksi secara langsung. Penelitian ini tidak hanya bersifat akademis, tetapi juga menghadirkan solusi praktis yang siap digunakan.

Dataset

Penelitian ini menggunakan dataset dari situs Kaggle yang memuat statistik pemain NBA musim 2024–2025. Dataset ini berisi data rinci dari setiap pertandingan, di mana tiap baris mewakili performa individu seorang pemain dalam satu laga. Informasi yang dikumpulkan mencakup berbagai aspek penting seperti menit bermain (MP), field goals (FG), poin (PTS), assist (AST), steal (STL), blok (BLK), nama tim (Tm), dan lawan tanding (Opp), yang kemudian dianalisis untuk melihat keterkaitannya dengan hasil pertandingan.

Sebagai data real-world yang berasal dari musim terkini, dataset ini mencerminkan tren dan dinamika terbaru di kompetisi NBA. Hal ini memberikan keunggulan bagi model yang dibangun karena dapat belajar dari kondisi permainan yang relevan. Volume data yang cukup besar juga memungkinkan model menangkap pola dari berbagai jenis pertandingan, baik saat menghadapi tim unggulan maupun tim yang lebih lemah, sehingga meningkatkan akurasi dan generalisasi prediksi.

Kode

Link Colab

IMPORT LIBRARY

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pickle

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

Import Library :
(1 & 2) Digunakan untuk manipulasi dan analisis data. (1) Untuk membaca dan mengolah tabel data (dataframe). (2) Untuk operasi matematis dan array. (3 & 4) Library untuk membuat visualisasi data. (5) Digunakan untuk menyimpan dan memuat model machine learning dalam bentuk file .pkl. (6) Untuk membagi dataset menjadi data pelatihan dan data pengujian. (7) Untuk preprocessing data (mengubah data kategorikal dan menormalkan fitur numerik). (8) Untuk mengatasi nilai kosong dalam data. (9 & 10) Algoritma klasifikasi utama yang digunakan dalam proyek. (11) Untuk evaluasi model dengan cross-validation. (12) Untuk mengukur performa model berdasarkan hasil prediksi.

LOAD DATASET

```
df = pd.read_csv("database_24_25.csv")
```

Baris kode ini berfungsi untuk memuat dataset NBA dari file bernama database_24_25.csv ke dalam sebuah DataFrame dengan nama df.

Dataset tersebut berisi data statistik pemain NBA selama musim 2024–2025. Setelah dimuat, data ini akan digunakan untuk proses eksplorasi, preprocessing, pembuatan model prediksi, serta evaluasi performa model.

Preprocessing

PREPROCESSING DATA

CEK MISSING VALUES

```
missing_values = df.isnull().sum()
```

FEATURE ENGINEERING

```
df_clean = df.copy()

df_clean['Res'] = df_clean['Res'].map({'W': 1, 'L': 0})

df_clean.drop(columns=['Player'], axis=1, inplace=True)
```

Digunakan untuk memeriksa apakah ada data yang kosong (null) di dalam dataset. Ini penting untuk mengetahui apakah perlu dilakukan penanganan seperti pengisian nilai atau penghapusan data.

Dataset disalin ke variabel baru df_clean agar data asli tetap aman.
Nilai kolom Res (hasil pertandingan) diubah dari 'W' dan 'L' menjadi 1 dan 0 — ini menjadi label untuk klasifikasi.
Kolom Player dihapus karena tidak berpengaruh langsung terhadap prediksi hasil pertandingan.

Preprocessing

ENCODING & NORMALISASI

```
le = LabelEncoder()
df_clean['Tm'] = le.fit_transform(df_clean['Tm'])
df_clean['Opp'] = le.fit_transform(df_clean['Opp'])

X = df_clean.drop('Res', axis=1)
y = df_clean['Res']

X = X.select_dtypes(exclude='object')

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

- Kolom 'Tm' (Tim) dan 'Opp' (Lawan) adalah teks, maka diubah ke bentuk angka menggunakan Label Encoding agar bisa dibaca oleh model.
- X adalah fitur (data input), y adalah label (target).
- select_dtypes(exclude='object') memastikan hanya kolom numerik yang digunakan sebagai fitur.
- Fitur numerik dinormalisasi menggunakan StandardScaler agar semua kolom berada dalam skala yang sama. Ini penting agar model tidak berat sebelah terhadap kolom dengan nilai besar.

SPLIT DATA

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

X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((13209, 22), (3303, 22), (13209,), (3303,))
```

- Dataset dibagi menjadi 80% untuk pelatihan dan 20% untuk pengujian.
- random_state=42 digunakan agar pembagian data tetap sama setiap kali dijalankan (reproducible).
- Mengecek ukuran data train dan test, memastikan pembagian data berjalan dengan benar.

Exploratory Data Analysis (EDA)

DISTRIBUSI HASIL PERTANDINGAN

```
plt.figure(figsize=(5, 4))
sns.countplot(x=y, hue=y, palette='viridis', legend=False)
plt.title("Distribusi Kemenangan vs Kekalahan")
plt.xlabel("Hasil (0 = Loss, 1 = Win)")
plt.ylabel("Jumlah")
plt.show()
```

Visualisasi ini menunjukkan jumlah data pertandingan menang (1) dan kalah (0). Hasil ini penting untuk mengetahui apakah data target (label) seimbang atau tidak. Jika terlalu tidak seimbang (misal terlalu banyak menang), model bisa bias.

RATA-RATA POIN

```
plt.figure(figsize=(6, 4))
sns.boxplot(x='Res', y='PTS', data=df_clean, hue='Res', palette='pastel', legend=False)
plt.title("Distribusi Poin Pemain berdasarkan Hasil Pertandingan")
plt.xlabel("Hasil (0 = Loss, 1 = Win)")
plt.ylabel("Points (PTS)")
plt.show()
```

Menggunakan boxplot untuk melihat penyebaran poin (PTS) yang dicetak oleh pemain, dipisah berdasarkan hasil pertandingan. Visualisasi ini membantu kita mengetahui apakah pemain cenderung mencetak poin lebih tinggi saat tim menang — dan biasanya hasilnya memang demikian.

DISTRIBUSI MENIT BERMAIN

```
plt.figure(figsize=(6, 4))
sns.histplot(df_clean['MP'], kde=True, bins=30, color='skyblue')
plt.title("Distribusi Menit Bermain Pemain (MP)")
plt.xlabel("Menit Bermain")
plt.ylabel("Jumlah")
plt.show()
```

Grafik ini menunjukkan distribusi jumlah menit bermain pemain dalam satu pertandingan. Dengan kurva KDE (kernel density estimation), kita bisa melihat seberapa sering pemain bermain dalam rentang menit tertentu (misalnya 20–35 menit paling umum).

Exploratory Data Analysis (EDA)

HUBUNGAN ANTARA MENIT BERMAIN & POIN

```
plt.figure(figsize=(6, 4))
sns.scatterplot(data=df_clean, x='MP', y='PTS', hue='Res', palette='Set1')
plt.title("Scatterplot Menit Bermain vs Poin")
plt.xlabel("Menit Bermain (MP)")
plt.ylabel("Poin (PTS)")
plt.legend(title='Hasil')
plt.show()
```

Scatterplot ini menunjukkan hubungan antara jumlah menit bermain (MP) dan jumlah poin yang dicetak (PTS). Data diberi warna berdasarkan hasil pertandingan (menang/kalah). Ini memberi gambaran apakah pemain yang bermain lebih lama cenderung mencetak lebih banyak poin, dan apakah itu berkorelasi dengan kemenangan.

KORELASI ANTAR FITUR

```
plt.figure(figsize=(12, 10))
correlation_matrix = df_clean.corr(numeric_only=True)
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap="coolwarm", square=True)
plt.title("Heatmap Korelasi antar Fitur Numerik")
plt.show()
```

Heatmap korelasi menampilkan hubungan antar fitur numerik dalam dataset. Korelasi tinggi antara dua fitur bisa berarti redundansi, dan korelasi kuat antara fitur dengan Res menunjukkan bahwa fitur tersebut relevan untuk prediksi.

RATA-RATA STATISTIK PENTING BERDASARKAN KEMENANGAN

```
features = ['FG', 'PTS', 'AST', 'STL', 'BLK']
mean_stats = df_clean.groupby('Res')[features].mean().T

plt.figure(figsize=(8, 5))
mean_stats.plot(kind='bar', colormap='viridis')
plt.title("Rata-Rata Statistik Pemain Berdasarkan Hasil Pertandingan")
plt.ylabel("Rata-rata")
plt.xticks(rotation=0)
plt.legend(title='Hasil (0=Loss, 1=Win)')
plt.tight_layout()
plt.show()
```

Diagram batang ini membandingkan rata-rata dari beberapa statistik penting (FG, PTS, AST, STL, BLK) antara pertandingan yang dimenangkan (1) dan yang kalah (0). Ini menunjukkan bahwa kemenangan sering diikuti oleh statistik yang lebih tinggi di semua fitur tersebut.

Contohnya:

- Jika PTS memiliki korelasi tinggi dengan Res, artinya semakin tinggi poin, semakin besar kemungkinan tim menang.
- Korelasi negatif antar fitur juga bisa membantu menghindari multikolinearitas.

Metodologi

DECISION TREE

```python
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)
y_pred_dt = dt_model.predict(X_test)

print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred_dt))
print("Classification Report - Decision Tree:\n", classification_report(y_test, y_pred_dt))
```

```
Decision Tree Accuracy: 0.6769603390856797
Classification Report - Decision Tree:
               precision    recall  f1-score   support

           0       0.67      0.68      0.68      1650
           1       0.68      0.67      0.67      1653

    accuracy                           0.68      3303
   macro avg       0.68      0.68      0.68      3303
weighted avg       0.68      0.68      0.68      3303
```

Decision Tree

- Decision Tree adalah model yang membagi data berdasarkan fitur-fitur penting dan membuat keputusan seperti pohon bercabang.
- Setelah model dilatih, dilakukan prediksi terhadap data uji (X_test).
- Hasil prediksi kemudian dibandingkan dengan label asli (y_test) untuk mengevaluasi akurasi.
- Digunakan untuk menampilkan akurasi, serta metrik seperti precision, recall, dan f1-score.

Metodologi

Random Forest

RANDOM FOREST

```python
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)

print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
print("Classification Report - Random Forest:\n", classification_report(y_test, y_pred_rf))
```

```
Random Forest Accuracy: 0.6612170753860127
Classification Report - Random Forest:
               precision    recall  f1-score   support

           0       0.65      0.69      0.67      1650
           1       0.67      0.64      0.65      1653

    accuracy                           0.66      3303
   macro avg       0.66      0.66      0.66      3303
weighted avg       0.66      0.66      0.66      3303
```

- Random Forest adalah gabungan dari banyak Decision Tree. Setiap pohon dilatih dengan data acak, dan hasilnya digabungkan melalui voting mayoritas.
- Lebih akurat dan tahan terhadap overfitting dibandingkan Decision Tree tunggal.

Hasil Evaluasi

CROSS VALIDATION SCORE

```python
cv_score_dt = cross_val_score(dt_model, X_scaled, y, cv=5)
cv_score_rf = cross_val_score(rf_model, X_scaled, y, cv=5)

print("CV Score Decision Tree:", cv_score_dt.mean())
print("CV Score Random Forest:", cv_score_rf.mean())
```

```
CV Score Decision Tree: 0.5843611510414058
CV Score Random Forest: 0.6395335408058272
```

- Cross-validation adalah cara evaluasi tambahan untuk mengukur seberapa stabil performa model di berbagai subset data.
- Skor rata-rata dari 5-fold cross-validation ditampilkan untuk Decision Tree dan Random Forest.
- Ini membantu melihat apakah model terlalu tergantung pada satu pembagian data atau tidak.

VISUALISASI CONFUSION MATRIX

```python
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
sns.heatmap(confusion_matrix(y_test, y_pred_dt), annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix - Decision Tree")
plt.xlabel("Predicted")
plt.ylabel("Actual")

plt.subplot(1, 2, 2)
sns.heatmap(confusion_matrix(y_test, y_pred_rf), annot=True, fmt='d', cmap='Greens')
plt.title("Confusion Matrix - Random Forest")
plt.xlabel("Predicted")
plt.ylabel("Actual")

plt.tight_layout()
plt.show()
```

Confusion Matrix menggambarkan jumlah prediksi benar dan salah :

- Kolom: hasil prediksi model
- Baris: label sebenarnya
- Angka diagonal: prediksi benar (True Positive & True Negative)
- Angka di luar diagonal: kesalahan prediksi

Grafik ditampilkan secara berdampingan agar bisa langsung dibandingkan.

Hasil Evaluasi

FEATURE IMPORTANCE

```
importances = rf_model.feature_importances_
feature_names = X.columns
sorted_idx = importances.argsort()[::-1]

plt.figure(figsize=(10, 6))
sns.barplot(x=importances[sorted_idx], y=feature_names[sorted_idx])
plt.title("Feature Importance - Random Forest")
plt.xlabel("Importance")
plt.ylabel("Feature")
plt.tight_layout()
plt.show()
```

Random Forest

- Random Forest menyediakan nilai pentingnya masing-masing fitur.
- Fitur dengan nilai tertinggi (misalnya: PTS, AST, STL) dianggap paling berpengaruh dalam menentukan kemenangan atau kekalahan tim.
- Visualisasi ini membantu mengetahui fitur mana yang paling relevan untuk prediksi.

Kesimpulan

- Random Forest memberikan akurasi dan stabilitas yang lebih baik dibandingkan Decision Tree.
- Visualisasi mendukung klaim ini: confusion matrix Random Forest menunjukkan prediksi yang lebih presisi, dan fitur penting dapat diidentifikasi dengan jelas.

Pra-Deployment

PRA DEPLOYMENT

```python
df_clean.to_csv("nba_cleaned_dataset.csv", index=False)
```

```python
with open("rf_model.pkl", "wb") as f:
    pickle.dump(rf_model, f)

with open("scaler.pkl", "wb") as f:
    pickle.dump(scaler, f)
```

```python
with open("label_encoder.pkl", "wb") as f:
    pickle.dump(le, f)
```

```python
print("Decision Tree - Train Accuracy:", dt_model.score(X_train, y_train))
print("Random Forest - Train Accuracy:", rf_model.score(X_train, y_train))
```

```
Decision Tree - Train Accuracy: 1.0
Random Forest - Train Accuracy: 1.0
```

- Dataset hasil preprocessing disimpan ke dalam file .csv.
- File ini bisa digunakan untuk dokumentasi, kebutuhan visualisasi di aplikasi, atau pelatihan ulang di masa depan.

- Model Random Forest disimpan ke file rf_model.pkl.
- Format .pkl (pickle) memungkinkan model digunakan kembali tanpa harus dilatih ulang.
- Scaler dan Label Encoder juga disimpan agar proses input di aplikasi bisa disesuaikan dengan preprocessing yang digunakan saat training.
- Ini penting karena input dari pengguna di aplikasi web nantinya harus melalui proses normalisasi dan encoding yang sama persis seperti saat model dilatih.

- Akurasi model dihitung pada data pelatihan.
- Ini berguna untuk mengecek apakah model overfitting (misalnya: terlalu bagus di train tapi buruk di test).
- Biasanya Random Forest menunjukkan akurasi tinggi pada data latih karena sifat ensemble-nya yang kuat.

Deployment

```python
import streamlit as st
import pickle
import numpy as np

with open("rf_model.pkl", "rb") as f:
    model = pickle.load(f)

with open("scaler.pkl", "rb") as f:
    scaler = pickle.load(f)

with open("label_encoder.pkl", "rb") as f:
    le = pickle.load(f)
```

- streamlit: digunakan untuk membangun dan menampilkan aplikasi web.
- pickle: untuk memuat ulang model, scaler, dan encoder yang sebelumnya disimpan.
- numpy: untuk mengubah input menjadi array numerik sebelum dikirim ke model.

- Model Random Forest (rf_model.pkl), Scaler (scaler.pkl), dan Label Encoder (label_encoder.pkl) dimuat ke dalam aplikasi.
- Ini penting agar input pengguna dapat diproses dengan cara yang sama seperti saat model dilatih.

Deployment

```python
st.title("Prediksi Hasil Pertandingan NBA (Menang atau Kalah) 🏀")
st.markdown("Masukkan data statistik pemain untuk memprediksi apakah tim akan menang (1) atau kalah (0).")

mp = st.slider("Menit Bermain (MP)", 0, 48, 30)
fg = st.number_input("Field Goals (FG)", 0.0, 30.0, 5.0)
pts = st.number_input("Points (PTS)", 0.0, 70.0, 20.0)
ast = st.number_input("Assists (AST)", 0.0, 20.0, 5.0)
stl = st.number_input("Steals (STL)", 0.0, 10.0, 1.0)
blk = st.number_input("Blocks (BLK)", 0.0, 10.0, 1.0)

all_teams = list(le.classes_)
team = st.selectbox("Nama Tim", all_teams, index=all_teams.index("LAL") if "LAL" in all_teams else 0)
opponent = st.selectbox("Lawan", all_teams, index=all_teams.index("BOS") if "BOS" in all_teams else 1)
```

- Tampilan aplikasi diberi judul dan deskripsi singkat.
- Pengguna diminta mengisi statistik pemain melalui form interaktif.
- Input mencakup: menit bermain (MP), field goals (FG), poin (PTS), assist (AST), steal (STL), dan blok (BLK).
- Dropdown menu digunakan untuk memilih nama tim dan lawan, sehingga meminimalkan kesalahan input teks.

Deployment

```python
if team in le.classes_:
    encoded_team = le.transform([team])[0]
else:
    st.warning(f"Tim '{team}' tidak dikenali. Pastikan sesuai dataset.")
    encoded_team = 0

if opponent in le.classes_:
    encoded_opp = le.transform([opponent])[0]
else:
    st.warning(f"Tim lawan '{opponent}' tidak dikenali. Pastikan sesuai dataset.")
    encoded_opp = 0
```

- Nama tim dan lawan dikonversi ke angka menggunakan LabelEncoder.
- Semua data input diskalakan dengan StandardScaler agar sesuai dengan format data yang dilatih oleh model.

```python
if st.button("Prediksi"):
    prediction = model.predict(scaled_input)[0]
    result = "Menang (1)" if prediction == 1 else "Kalah (0)"
    st.subheader(f"Hasil Prediksi: **{result}**")
```

```python
input_data = [
    encoded_team,   # Tm
    encoded_opp,    # Opp
    0,              # Res
    mp,             # MP
    fg,             # FG
    0,              # FGA
    0.0,            # FG%
    0,              # 3P
    0,              # 3PA
    0.0,            # 3P%
    0,              # FT
    0,              # FTA
    0.0,            # FT%
    0,              # ORB
    0,              # DRB
    0,              # TRB
    ast,            # AST
    stl,            # STL
    blk,            # BLK
    0,              # TOV
    0,              # PF
    pts,            # PTS
]

input_array = np.array(input_data).reshape(1, -1)
scaled_input = scaler.transform(input_array)
```

Ketika tombol "Prediksi" ditekan, input dikirim ke model, dan hasil prediksi dikembalikan. Hasil ditampilkan secara langsung kepada pengguna:

- “Menang (1)” jika model memprediksi kemenangan.
- “Kalah (0)” jika model memprediksi kekalahan.

Tampilan Aplikasi

Aplikasi ini memungkinkan siapa pun untuk:

- Memasukkan statistik pertandingan NBA secara manual.
- Mendapatkan hasil prediksi apakah tim tersebut akan menang atau kalah berdasarkan model Random Forest.
- Aplikasi bersifat interaktif, responsif, dan bisa dijalankan lokal atau online (seperti di Streamlit Cloud).

Referensi

Kaggle : NBA Dataset

About NBA

Source Code : Github

TERIMA KASIH

Mohon maaf atas kesalahan dan kekurangan selama presentasi berlangsung.