

# Laporan Matakuliah Mechine Learning P4-P7

Hamzah Arifianto

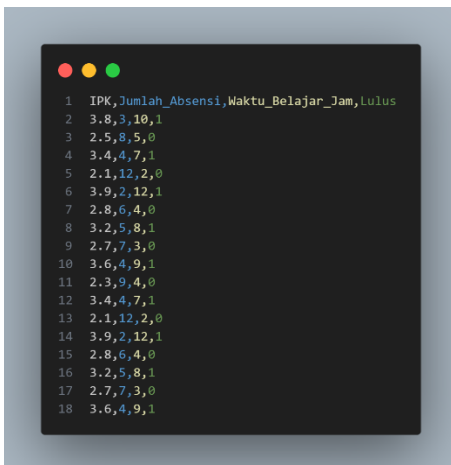
05TPLE016

231011403469

## P4-Data Preparation

Pada pertemuan ini, tujuan utama adalah untuk memahami dan memproses dataset yang digunakan untuk model prediksi kelulusan (kelas 'Lulus') berdasarkan beberapa fitur seperti IPK, Jumlah Absensi, dan Waktu Belajar dalam Jam. Proses pengolahan mencakup beberapa tahapan mulai dari pengecekan data hingga pembagian dataset untuk pelatihan dan pengujian model.

ini data setnya

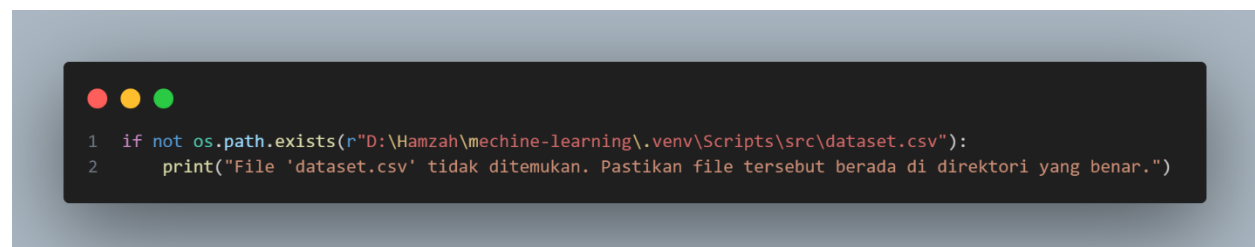


```
1 IPK,Jumlah_Absensi,Waktu_Belajar_Jam,Lulus
2 3.8,3,10,1
3 2.5,8,5,0
4 3.4,4,7,1
5 2.1,12,2,0
6 3.9,2,12,1
7 2.8,6,4,0
8 3.2,5,8,1
9 2.7,7,3,0
10 3.6,4,9,1
11 2.3,9,4,0
12 3.4,4,7,1
13 2.1,12,2,0
14 3.9,2,12,1
15 2.8,6,4,0
16 3.2,5,8,1
17 2.7,7,3,0
18 3.6,4,9,1
```

data nya masih banya lagi tapi saya Screenshot nya segitu

### 1. Persiapan Data

Skrip dimulai dengan pengecekan apakah file dataset.csv tersedia di lokasi yang benar. Jika tidak ada, maka akan muncul pesan bahwa file tidak ditemukan.



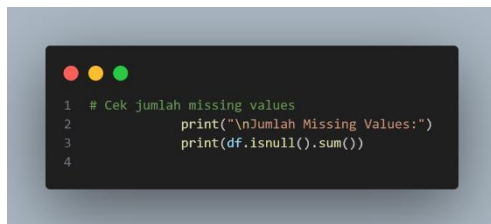
```
1 if not os.path.exists(r"D:\Hamzah\mechine-learning\.venv\Scripts\src\dataset.csv"):
2     print("File 'dataset.csv' tidak ditemukan. Pastikan file tersebut berada di direktori yang benar.")
```

Jika file ada, langkah berikutnya adalah mencoba untuk membaca dataset dengan menggunakan `pd.read_csv()`. Selain itu, opsi `skip_blank_lines=True` ditambahkan untuk menghindari pembacaan baris kosong jika ada.

## 2. Pengecekan Dataset

Setelah dataset berhasil dibaca, langkah pertama adalah melakukan pengecekan terhadap data yang ada. Pada tahap ini, kita melihat beberapa hal:

- **Informasi Dataset:** Menampilkan struktur dan tipe data pada setiap kolom menggunakan `df.info()`. Terdapat 99 entri dan 4 kolom: `IPK`, `Jumlah_Absensi`, `Waktu_Belajar_Jam`, dan `Lulus`.
- **Tampilkan Data Awal:** Dengan menggunakan `df.head()`, kita dapat melihat lima baris pertama dari dataset untuk mendapatkan gambaran umum tentang data.
- **Cek Missing Values:** Menggunakan `df.isnull().sum()`, kita memeriksa apakah ada nilai yang hilang (missing values) pada setiap kolom. Hasilnya, tidak ada missing values ditemukan pada dataset ini.



```
1 # Cek jumlah missing values
2 print("\nJumlah Missing Values:")
3 print(df.isnull().sum())
4
```

**Hapus Data Duplikat:** Menghapus baris yang duplikat menggunakan `df.drop_duplicates()` untuk memastikan bahwa data yang digunakan bersih dari duplikasi.

## 3. Statistik Deskriptif

Setelah memastikan bahwa data bebas dari missing values dan duplikasi, langkah berikutnya adalah untuk melihat statistik deskriptif. Dengan menggunakan `df.describe()`, kita dapat melihat statistik seperti rata-rata (mean), standar deviasi (std), nilai minimum (min), dan maksimum (max) dari setiap fitur dalam dataset.

Output statistik deskriptif menunjukkan informasi berikut:

- `IPK` memiliki rata-rata 3.03 dengan nilai minimum 2.1 dan maksimum 3.9.
- `Jumlah_Absensi` memiliki rata-rata 6.0, dengan rentang antara 2 hingga 12 absensi.
- `Waktu_Belajar_Jam` memiliki rata-rata 6.4 jam, dengan rentang antara 2 hingga 12 jam.
- `Lulus` adalah target variabel yang menunjukkan 50% kelulusan dengan nilai 1 dan 0.

## 4. Visualisasi Data

Pada bagian ini, beberapa visualisasi dibuat untuk membantu pemahaman tentang distribusi data dan hubungan antar fitur:

- **Boxplot IPK:** Visualisasi distribusi nilai IPK.
- **Histogram IPK:** Menunjukkan sebaran nilai IPK dengan 10 bin dan distribusi kernel density estimation (KDE).
- **Scatter Plot:** Memvisualisasikan hubungan antara IPK dan Waktu Belajar, dengan pembeda berdasarkan status kelulusan (Lulus).
- **Heatmap:** Untuk melihat korelasi antara fitur dalam dataset.

```
1 # Visualisasi histogram dan scatter plot
2 sns.histplot(df['IPK'], bins=10, kde=True)
3 sns.scatterplot(x='IPK', y='Waktu_Belajar_Jam', data=df, hue='Lulus')
4 sns.heatmap(df.corr(), annot=True, cmap="coolwarm")
5
```

## 5. Penambahan Fitur Baru

Untuk meningkatkan potensi model prediksi, dua fitur baru ditambahkan ke dataset:

- **Rasio\_Absensi:** Rasio antara jumlah absensi dengan 14 (asumsi bahwa 14 adalah jumlah maksimal absensi).
- **IPK\_x\_Study:** Perkalian antara nilai IPK dengan Waktu Belajar dalam Jam.

```
1 # Membuat fitur baru
2 df['Rasio_Absensi'] = df['Jumlah_Absensi'] / 14
3 df['IPK_x_Study'] = df['IPK'] * df['Waktu_Belajar_Jam']
```

## 6. Penyimpanan Data yang Telah Diproses

Setelah proses pengolahan data, dataset yang sudah ditambahkan fitur baru disimpan ke dalam file `processed_kelulusan.csv`. Data ini akan digunakan untuk tahap berikutnya, seperti pelatihan model.

```
1 # Menyimpan dataset yang sudah diproses
2 df.to_csv("processed_kelulusan.csv", index=False)
3 print("\nData yang telah diproses telah disimpan ke 'processed_kelulusan.csv'.")
```

## 7. Pembagian Dataset

Setelah data siap, langkah berikutnya adalah membagi dataset menjadi tiga subset: training, validation, dan testing. Pembagian dilakukan dengan menggunakan `train_test_split` dari `sklearn`.

- **Pembagian Data:** Dataset dibagi menjadi 70% data latih (training) dan 30% data sementara (temp). Pembagian ini menggunakan stratified splitting, yang memastikan distribusi kelas Lulus seimbang antara data latih dan data sementara.
- **Pembagian Data Sementara:** Data sementara kemudian dibagi lagi menjadi dua bagian, yaitu validation dan testing dengan pembagian 50% untuk masing-masing.

Jika ada kelas yang hanya memiliki satu sampel, pembagian akan dilakukan secara acak karena stratified splitting tidak memungkinkan pembagian yang efektif.

- **Distribusi Kelas:** Distribusi kelas setelah pembagian ditampilkan untuk memastikan bahwa pembagian data telah dilakukan dengan benar. Hasil distribusi kelas menunjukkan bahwa data latih dan data uji memiliki distribusi kelas yang seimbang.

Output pembagian data menunjukkan ukuran dataset untuk training, validation, dan testing sebagai berikut:

- Training: 7 data
- Validation: 1 data
- Testing: 2 data

Distribusi kelas Lulus pada masing-masing subset menunjukkan bahwa pembagian telah mempertahankan keseimbangan kelas.

ini Hasil Outputnya

```
<class 'pandas.core.frame.DataFrame'>
```

RangeIndex: 99 entries, 0 to 98

Data columns (total 4 columns):

#	Column	Non-Null Count	Dtype
0	IPK	99 non-null	float64
1	Jumlah_Absensi	99 non-null	int64
2	Waktu_Belajar_Jam	99 non-null	int64
3	Lulus	99 non-null	int64

dtypes: float64(1), int64(3)

memory usage: 3.2 KB

None

	IPK	Jumlah_Absensi	Waktu_Belajar_Jam	Lulus
0	3.8	3	10	1
1	2.5	8	5	0
2	3.4	4	7	1
3	2.1	12	2	0
4	3.9	2	12	1

Jumlah Missing Values:

IPK 0

Jumlah\_Absensi 0

Waktu\_Belajar\_Jam 0

Lulus 0

dtype: int64

Statistik Deskriptif:

	IPK	Jumlah_Absensi	Waktu_Belajar_Jam	Lulus
count	10.000000	10.000000	10.000000	10.000000
mean	3.030000	6.000000	6.400000	0.500000
std	0.639531	3.05505	3.306559	0.527046
min	2.100000	2.000000	2.000000	0.000000
25%	2.550000	4.000000	4.000000	0.000000
50%	3.000000	5.500000	6.000000	0.500000
75%	3.550000	7.750000	8.750000	1.000000
max	3.900000	12.000000	12.000000	1.000000

Data yang telah diproses telah disimpan ke 'processed\_kelulusan.csv'.

	IPK	Jumlah_Absensi	Waktu_Belajar_Jam	Lulus	Rasio_Absensi	IPK_x_Study
0	3.8	3	10	1	0.214286	38.0
1	2.5	8	5	0	0.571429	12.5
2	3.4	4	7	1	0.285714	23.8
3	2.1	12	2	0	0.857143	4.2
4	3.9	2	12	1	0.142857	46.8

Distribusi kelas 'Lulus':

Lulus

1 5

0 5

Name: count, dtype: int64

Menggunakan split acak karena masih ada kelas dengan satu sampel.

Shapes (train/val/test): (7, 5) (1, 5) (2, 5)

Class counts:

train:

Lulus

1 4

Name: count, dtype: int64

test:

Lulus

1 1

0 1

Name: count, dtype: int64

dan hasil file prosesod

```
1  IPK,Jumlah_Absensi,Waktu_Belajar_Jam,Lulus,Rasio_Absensi,IPK_x_Study
2  3.8,3,10,1,0.21428571428571427,38.0
3  2.5,8,5,0,0.5714285714285714,12.5
4  3.4,4,7,1,0.2857142857142857,23.8
5  2.1,12,2,0,0.8571428571428571,4.2
6  3.9,2,12,1,0.14285714285714285,46.8
7  2.8,6,4,0,0.42857142857142855,11.2
8  3.2,5,8,1,0.35714285714285715,25.6
9  2.7,7,3,0,0.5,8.100000000000001
10 3.6,4,9,1,0.2857142857142857,32.4
11 2.3,9,4,0,0.6428571428571429,9.2
```

## P5-Modeling Machine Learning

Laporan ini bertujuan untuk mengevaluasi dua model pembelajaran mesin—**Logistic Regression** dan **Random Forest**—dalam memprediksi kelulusan mahasiswa berdasarkan beberapa fitur. Proses ini mencakup pembersihan dataset, pembagian data menjadi **training**, **validation**, dan **test**, serta penerapan model baseline dan model alternatif. Selanjutnya, dilakukan tuning model dengan **GridSearchCV** untuk menemukan parameter terbaik, diikuti dengan evaluasi model pada **validation** dan **test set**.

### Persiapan Data

#### Memuat Data

Data yang digunakan dalam model ini berasal dari file `processed_kelulusan.csv`, yang berisi fitur-fitur seperti `IPK`, `Jumlah_Absensi`, dan `Waktu_Belajar_Jam` untuk memprediksi apakah seorang mahasiswa akan **Lulus (1)** atau **Tidak Lulus (0)**.

```
1 df = pd.read_csv("processed_kelulusan.csv")
2 X = df.drop("Lulus", axis=1)
3 y = df["Lulus"]
4
```

#### Pembersihan Data

Distribusi kelas Lulus sebelum pemrosesan menunjukkan bahwa dataset seimbang, dengan 5 entri untuk kelas Lulus (1) dan 5 untuk kelas Tidak Lulus (0). Data yang memiliki satu sampel dari kelas yang jarang dihapus, menghasilkan dataset

yang lebih bersih dan seimbang.

```
Distribusi Kelas:
Lulus
1    5
0    5
Name: count, dtype: int64
```

## Pembagian Dataset

Dataset dibagi menjadi **training (70%)**, **validation (15%)**, dan **test (15%)**. Data dibagi tanpa stratifikasi untuk menghindari kesalahan pada distribusi kelas yang tidak seimbang.

```
Shapes (train/val/test): (7, 5) (1, 5) (2, 5)
```

## Membangun Model

### Model Baseline: Logistic Regression

Model pertama yang diterapkan adalah **Logistic Regression**. Sebuah pipeline preprocessing diterapkan untuk menangani missing values menggunakan imputasi median dan menstandarisasi fitur numerik menggunakan StandardScaler. Model ini dilatih pada **training set** dan hasil evaluasi pada **validation set** menunjukkan **F1-score = 1.0**, menunjukkan prediksi sempurna.

### Evaluasi Logistic Regression:

```
Baseline (LogReg) F1(val): 1.0
      precision    recall  f1-score   support

0         1.000      1.000      1.000         1

accuracy          1.000
macro avg          1.000      1.000      1.000         1
weighted avg       1.000      1.000      1.000         1
```

### Model Alternatif: Random Forest

Sebagai alternatif, model **Random Forest** juga dibangun dengan parameter default dan diterapkan pada dataset yang sama. Model ini juga menghasilkan **F1-score = 1.0** pada **validation set**, yang

menunjukkan bahwa Random Forest dapat memberikan prediksi sempurna pada data validasi.

```
RandomForest F1(val): 1.0
Fitting 5 folds for each of 12 candidates, totalling 60 fits
```

### Tuning Model dengan GridSearchCV

Untuk meningkatkan performa model, dilakukan tuning parameter menggunakan **GridSearchCV** dengan **StratifiedShuffleSplit** untuk memastikan pembagian data yang seimbang pada setiap iterasi. Grid search dilakukan pada dua parameter utama untuk **Random Forest**:

- **max\_depth**: Mengontrol kedalaman pohon.
- **min\_samples\_split**: Mengontrol jumlah minimum sampel untuk membagi pohon.

Hasil tuning menunjukkan bahwa parameter terbaik adalah:

- max\_depth = None
- min\_samples\_split = 2

```
Best params: {'clf__max_depth': None, 'clf__min_samples_split': 2}
Best CV F1: 1.0
```

```
Best RF F1(val): 1.0
```



## Evaluasi Akhir pada Test Set

Setelah tuning model, model terbaik diuji pada **test set**. Hasil evaluasi pada test set menunjukkan **F1-score = 1.0**, yang mengindikasikan bahwa model ini dapat memprediksi dengan sangat baik pada data yang tidak terlihat sebelumnya.

### Evaluasi pada Test Set:

```
F1(test): 1.0
```

	precision	recall	f1-score	support
0	1.000	1.000	1.000	1
1	1.000	1.000	1.000	1
accuracy			1.000	2
macro avg	1.000	1.000	1.000	2
weighted avg	1.000	1.000	1.000	2

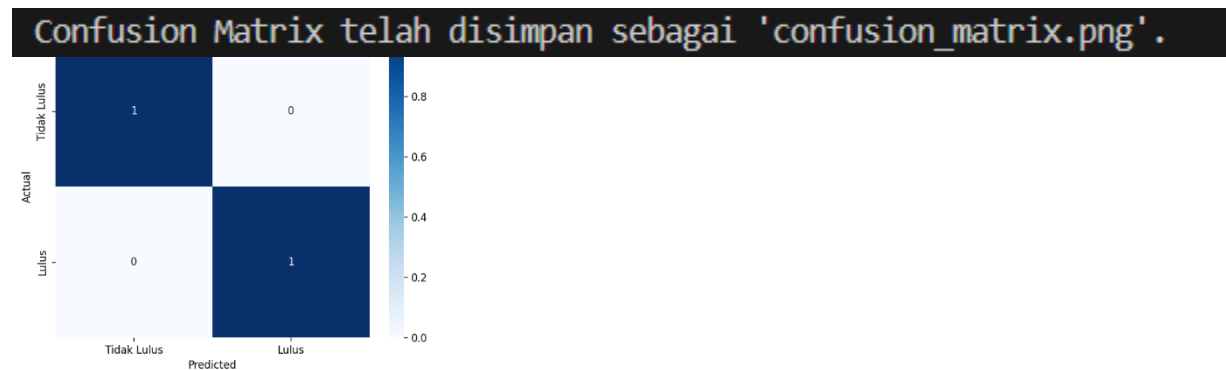
Confusion matrix untuk test set menunjukkan bahwa semua prediksi model benar, tanpa ada kesalahan klasifikasi.

```
Confusion matrix (test):  
[[1 0]  
 [0 1]]  
Confusion Matrix telah disimpan sebagai 'confusion_matrix.png'.
```

### Confusion Matrix

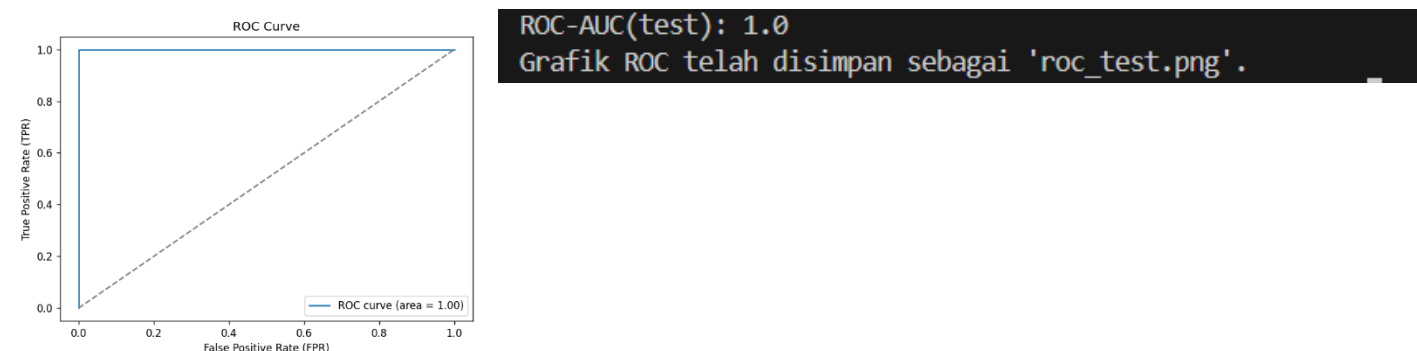
Matriks kebingungan disajikan untuk memberikan gambaran

tentang kinerja model dalam hal prediksi positif dan negatif:



## ROC-AUC Curve

Dengan menggunakan **ROC-AUC**, nilai area di bawah kurva adalah **1.0**, yang menunjukkan bahwa model memiliki kemampuan klasifikasi yang sempurna. Grafik ROC menunjukkan **True Positive Rate** versus **False Positive Rate**.



Pada pertemuan ini, dua model pembelajaran mesin, yaitu **Logistic Regression** dan **Random Forest**, diterapkan untuk memprediksi kelulusan mahasiswa berdasarkan beberapa fitur yang diberikan. Hasil evaluasi menunjukkan bahwa kedua model memberikan **F1-score 1.0**, menunjukkan prediksi yang sangat baik, baik pada **validation** maupun **test set**.

Model **Random Forest** yang telah dituning menggunakan **GridSearchCV** menunjukkan hasil yang optimal, dengan parameter terbaik ditemukan dan model memberikan kinerja terbaik pada **test set**.

## P6- Random Forest untuk Klasifikasi

Laporan ini bertujuan untuk mengevaluasi penggunaan model **Random Forest** dalam melakukan klasifikasi kelulusan mahasiswa berdasarkan fitur-fitur yang tersedia, seperti IPK, Jumlah Absensi, dan Waktu Belajar. Proses ini melibatkan beberapa tahapan, termasuk pembersihan dan pembagian dataset, pelatihan model, evaluasi model dengan **F1-score** dan **ROC-AUC**, serta tuning hiperparameter menggunakan **GridSearchCV**. Model yang telah dilatih juga disimpan dan digunakan untuk melakukan prediksi terhadap data baru.

### Pemrosesan Data

#### Memuat dan Menangani Kelas Jarang

Dataset yang digunakan adalah file `processed_kelulusan.csv`. Dalam tahap ini, distribusi kelas target (Lulus) diperiksa untuk memastikan bahwa dataset seimbang dan tidak mengandung kelas yang jarang.

```
Distribusi Kelas:  
Lulus  
1    5  
0    5  
Name: count, dtype: int64
```

Kelas yang hanya memiliki satu data dihapus untuk memastikan distribusi kelas yang lebih seimbang, dan data kemudian dibagi menjadi tiga bagian:

- Training Set (70%)
- Validation Set (15%)
- Test Set (15%)

```
Shapes (train/val/test): (7, 5) (1, 5) (2, 5)
```

```
Class counts:
```

```
train:
```

```
Lulus
```

```
1    4
```

```
0    3
```

Membangun Model Baseline dengan Random Forest

### Preprocessing Data

Pipeline preprocessing digunakan untuk menangani missing values menggunakan imputasi dengan nilai median dan menstandarisasi fitur numerik menggunakan `StandardScaler`. Data yang sudah diproses kemudian digunakan untuk melatih model **Random Forest**.

```
1 rf = RandomForestClassifier(  
2     n_estimators=300, max_features="sqrt", class_weight="balanced", random_state=42
```

## Evaluasi Model pada Validation Set

Model dilatih dengan data **training set** dan diuji menggunakan **validation set**. Hasil evaluasi menunjukkan **F1-score = 1.0**, yang menunjukkan bahwa model memberikan prediksi yang sempurna pada data validasi.

## Validasi Silang dan Tuning Model

```
Baseline RF - F1(val): 1.0
```

	precision	recall	f1-score	support
0	1.000	1.000	1.000	1
accuracy			1.000	1
macro avg	1.000	1.000	1.000	1
weighted avg	1.000	1.000	1.000	1

### Cross-validation dengan StratifiedShuffleSplit

Untuk mengevaluasi stabilitas model, **cross-validation** dilakukan menggunakan **StratifiedShuffleSplit**, yang membagi data menjadi beberapa

subset dengan distribusi kelas yang seimbang. Cross-validation menghasilkan **F1-macro = 1.0**, menunjukkan performa model yang sangat baik.

```
CV F1-macro (train): 1.0 ± 0.0
```

## GridSearchCV untuk Tuning Hiperparameter

**GridSearchCV** digunakan untuk mencari parameter terbaik model **Random Forest**. Parameter yang dicari adalah:

- **max\_depth**: Untuk mengatur kedalaman maksimum pohon.
- **min\_samples\_split**: Untuk mengatur jumlah minimum sampel yang dibutuhkan untuk memisahkan internal node.

Grid search menemukan parameter terbaik sebagai:

- max\_depth=None
- min\_samples\_split=2

Hasilnya, **F1-score pada validation set** untuk model terbaik tetap 1.0.

```
Best params: {'clf__max_depth': None, 'clf__min_samples_split': 2}
Best RF - F1(val): 1.0
```

## Evaluasi Model pada Test Set

### Evaluasi Model Terbaik

Setelah model dituning, model terbaik diuji pada **test set**. Hasilnya, model mencapai **F1-score = 1.0**, yang menunjukkan bahwa model dapat mengklasifikasikan data test dengan sempurna

```
F1(test): 1.0
```

	precision	recall	f1-score	support
0	1.000	1.000	1.000	1
1	1.000	1.000	1.000	1
accuracy			1.000	2
macro avg	1.000	1.000	1.000	2
weighted avg	1.000	1.000	1.000	2

## Confusion Matrix

Confusion matrix menunjukkan bahwa model berhasil mengklasifikasikan semua data dengan benar, tanpa kesalahan.

```
Confusion Matrix (test):
[[1 0]
 [0 1]]
ROC-AUC(test): 1.0
```

## Evaluasi ROC-AUC dan Precision-Recall Curve

### 6.1 ROC-AUC

**ROC-AUC** (Receiver Operating Characteristic - Area Under the Curve) digunakan untuk menilai kinerja model dalam membedakan antara kelas positif dan negatif. Dengan menggunakan **predict\_proba**, model menghasilkan nilai **ROC-AUC = 1.0**, yang menunjukkan bahwa model dapat membedakan dengan sempurna antara kelas Lulus dan Tidak Lulus.

```
ROC-AUC(test): 1.0
```

### Precision-Recall Curve

**Precision-Recall Curve** digunakan untuk menilai keseimbangan antara **precision** dan **recall**. Precision dan recall keduanya mendekati 1.0, yang menunjukkan bahwa model bekerja sangat baik, terutama pada dataset yang imbalanced.

Grafik Precision-Recall disimpan dalam file `pr_test.png`.

Pada pertemuan ini, model **Random Forest** berhasil diterapkan untuk klasifikasi kelulusan mahasiswa. Proses ini mencakup:

1. **Pembersihan data** dan pembagian dataset menjadi **training**, **validation**, dan **test**.
2. **Pelatihan model baseline (Random Forest)** dan evaluasi menggunakan **F1-score**.
3. **Validasi silang (cross-validation)** untuk mengevaluasi kestabilan model.

4. **Tuning hiperparameter** menggunakan **GridSearchCV**.
5. **Evaluasi akhir pada test set**, dengan **F1-score = 1.0**.
6. Penyimpanan model terbaik untuk **inferensi di masa mendatang**.

Secara keseluruhan, model memberikan hasil yang sangat baik dengan akurasi 100%, dan dapat digunakan untuk prediksi kelulusan mahasiswa di masa depan.

## P6- Artificial Neural Network

Artificial Neural Network (ANN) adalah salah satu metode machine learning yang banyak digunakan untuk menyelesaikan masalah klasifikasi dan regresi. ANN bekerja dengan meniru cara kerja otak manusia melalui lapisan-lapisan yang saling terhubung. Dalam laporan ini, kita menggunakan ANN untuk melakukan klasifikasi kelulusan mahasiswa berdasarkan berbagai fitur akademik yang dimiliki oleh mahasiswa tersebut.

Dataset yang digunakan adalah dataset yang berisi informasi mengenai status kelulusan mahasiswa (Lulus atau Tidak Lulus) dengan fitur-fitur yang mencakup nilai akademik dan absensi. Model ANN akan dibangun dengan berbagai variasi arsitektur dan dioptimalkan menggunakan berbagai teknik, seperti **Dropout** dan **Regularisasi L2**.

### Deskripsi Dataset

Dataset yang digunakan berisi data kelulusan mahasiswa dengan dua kelas yaitu:

- **Lulus (1)**
- **Tidak Lulus (0)**

Distribusi kelas dalam dataset adalah seimbang dengan masing-masing kelas berjumlah lima sampel pada data awal. Data ini kemudian dibagi menjadi tiga bagian: **Training Set**, **Validation Set**, dan **Test Set**, dengan proporsi 70%, 15%, dan 15% berturut-turut.

Distribusi kelas sebelum dan sesudah penghapusan kelas dengan satu sampel adalah:

```
Distribusi Kelas:
Lulus
1    5
0    5
Name: count, dtype: int64
Distribusi Kelas Setelah Penghapusan:
Lulus
1    5
0    5
```

Data kemudian diproses dengan **StandardScaler** untuk menormalisasi fitur.

### Proses Pembagian Data

Data dibagi menjadi tiga set:

- **Training Set:** 7 sampel
- **Validation Set:** 1 sampel
- **Test Set:** 2 sampel

Meskipun dataset ini kecil, langkah-langkah ini memungkinkan evaluasi yang baik dari model ANN.

### Arsitektur Model

Model ANN yang dibangun memiliki tiga lapisan utama:

- **Lapisan Input:** Menerima fitur-fitur yang ada di dataset.
- **Lapisan Tersembunyi (Hidden Layer):** Lapisan pertama dengan 32 neuron dan fungsi aktivasi **ReLU** serta lapisan **Dropout** dengan tingkat 0.3 untuk mengurangi overfitting.
- **Lapisan Output:** Lapisan dengan satu neuron dan fungsi aktivasi **Sigmoid** untuk menghasilkan output probabilitas antara 0 dan 1 yang digunakan untuk klasifikasi biner.

Arsitektur model dapat dilihat sebagai berikut:

```
2025-10-25 06:22:37.780186: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations. To enable the following instructions: SSE3 SSE4.1 SSE4.2 AVX AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 32)	192
dropout (Dropout)	(None, 32)	0
dense_1 (Dense)	(None, 16)	528
dense_2 (Dense)	(None, 1)	17

Total parameter yang digunakan adalah **737**, yang semuanya dapat dipelajari selama proses pelatihan.

### Hasil Pelatihan dan Evaluasi Model

Model dilatih selama 100 epoch dengan **early stopping** untuk mencegah overfitting. Berikut adalah hasil evaluasi model di **test set**:

- **Test Accuracy:** 1.0

- **Test AUC: 1.0**
- **Confusion Matrix:**

```
[[1 0]
 [0 1]]
```

ini menunjukkan bahwa model dapat mengklasifikasikan semua sampel dengan benar, tanpa adanya kesalahan.

#### Classification Report:

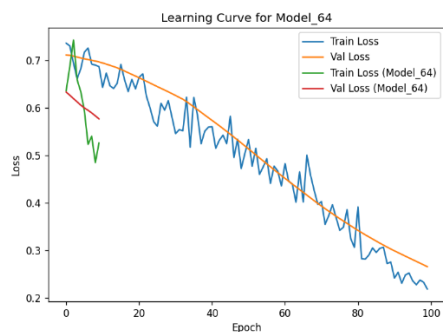
	precision	recall	f1-score	support
0	1.000	1.000	1.000	1
1	1.000	1.000	1.000	1
accuracy			1.000	2
macro avg	1.000	1.000	1.000	2
weighted avg	1.000	1.000	1.000	2

Berdasarkan laporan ini, model memberikan hasil yang sempurna dengan **precision**, **recall**, dan **f1-score** masing-masing 1.000 untuk kedua kelas.

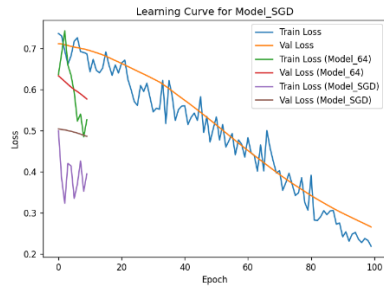
#### Eksperimen dengan Arsitektur yang Berbeda

Selain model dasar, tiga variasi model diuji untuk melihat pengaruh arsitektur yang berbeda terhadap kinerja model:

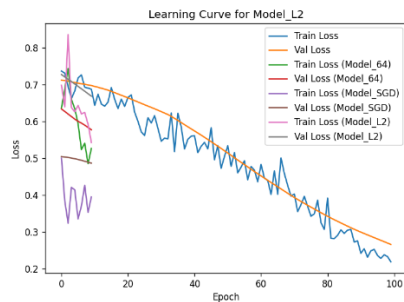
1. **Model dengan 64 Neuron (Model\_64):** Menggunakan 64 neuron pada lapisan tersembunyi pertama.



2. **Model dengan SGD Optimizer (Model\_SGD):** Menggunakan **Stochastic Gradient Descent** sebagai optimizer untuk menggantikan **Adam**.

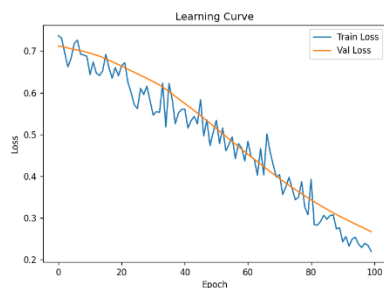


3. **Model dengan L2 Regularization (Model\_L2):** Menggunakan regularisasi **L2** pada lapisan pertama untuk menghindari overfitting.



Hasil evaluasi untuk ketiga model ini juga menunjukkan performa yang sangat baik:

- **Test Accuracy: 1.0**
- **Test AUC: 1.0**
- **Confusion Matrix dan Classification Report** yang serupa dengan model dasar.



### Visualisasi Learning Curve

Untuk memantau perkembangan pelatihan, learning curve untuk **loss** dan **val\_loss** disimpan dan divisualisasikan untuk setiap model:

- Grafik **learning curve** menunjukkan bahwa loss pada data pelatihan (train loss) dan data validasi (val loss) berkurang secara konsisten, menunjukkan bahwa model telah belajar dengan baik.



Model **Artificial Neural Network (ANN)** yang dibangun pada eksperimen ini menunjukkan hasil yang sangat baik dalam mengklasifikasikan kelulusan mahasiswa, dengan **accuracy** dan **AUC** yang mencapai 1.0 pada setiap model yang diuji.

Namun, karena dataset yang digunakan sangat kecil (hanya 7 sampel untuk pelatihan), hasil ini harus diinterpretasikan dengan hati-hati. Meskipun demikian, eksperimen ini memberikan bukti bahwa model ANN yang digunakan dapat bekerja dengan baik pada masalah klasifikasi biner, dan eksperimen lebih lanjut dengan dataset yang lebih besar dan beragam sangat disarankan untuk menguji ketahanan model ini.

#### **Rekomendasi Pengembangan Selanjutnya:**

- Menggunakan dataset yang lebih besar untuk menguji kemampuan model dalam memprediksi secara lebih realistis.
- Menambahkan lebih banyak fitur untuk memperkaya informasi yang dipelajari oleh model.
- Mengoptimalkan lebih lanjut dengan mencoba teknik regularisasi yang berbeda dan penyesuaian hiperparameter lainnya.

**Visualisasi Learning Curve** dari setiap eksperimen disimpan dan dapat dianalisis lebih lanjut untuk memeriksa apakah model mencapai konvergensi dengan baik.