

KLASIFIKASI

NADILA IMAARAH

SAINS DATA TERAPAN A

332360015



APA ITU KLASIFIKASI ?

Klasifikasi adalah pengelompokkan data atau objek baru ke dalam kelas atau label berdasarkan atribut-atribut tertentu. Klasifikasi bertujuan untuk memprediksi kelas dari suatu objek yang tidak diketahui sebelumnya. Klasifikasi terdiri dari tiga tahap, yaitu pembangunan model, penerapan model, dan evaluasi. Pembangunan model adalah membangun model menggunakan data latih yang telah memiliki atribut dan kelas. Kemudian, data-data tersebut diterapkan untuk menentukan kelas dari data atau objek yang baru. Setelah itu, data dievaluasi untuk melihat tingkat akurasi dari pembangunan dan penerapan model terhadap data baru

TOPIK OVERVIEW

00

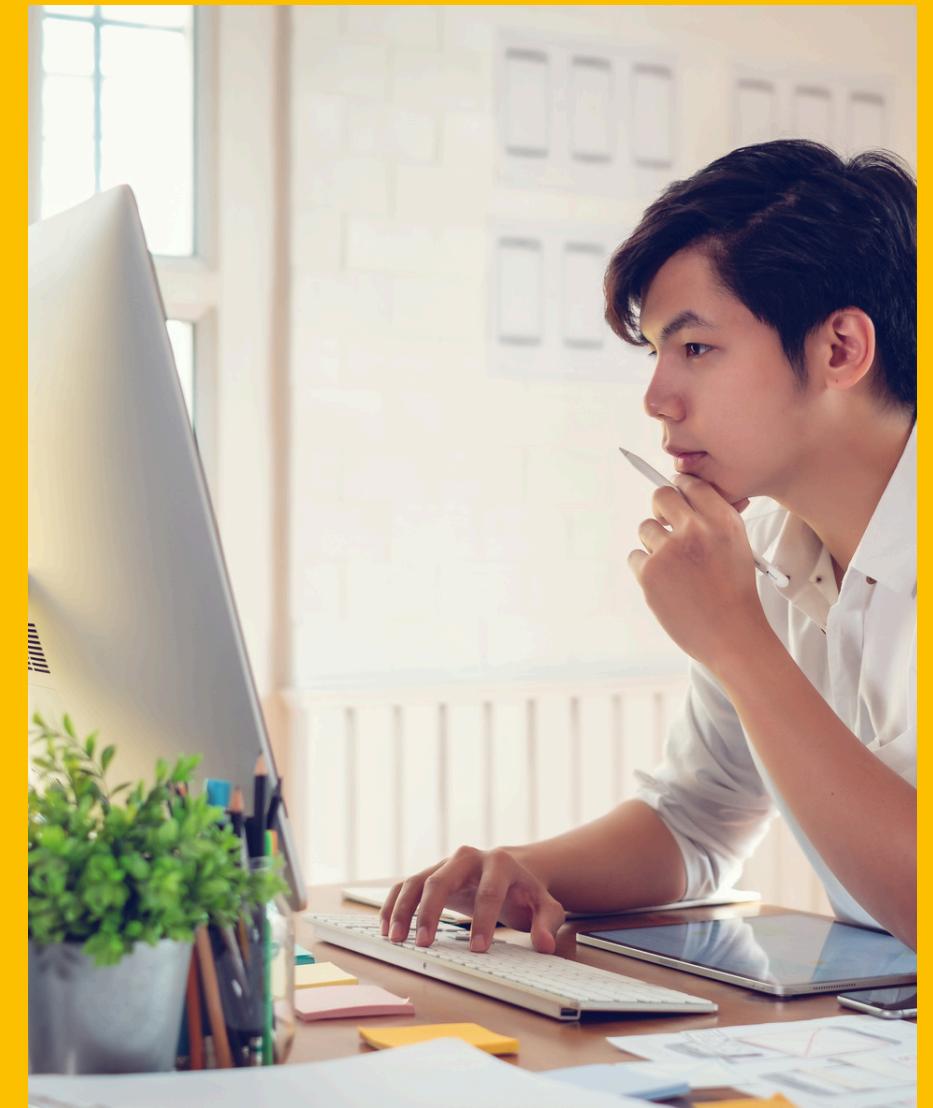
Latihan

01

Membaca Dataset
Titanic

02

Membagi Dataset
menjadi Data Train
dan Data Test



TOPIK OVERVIEW

03

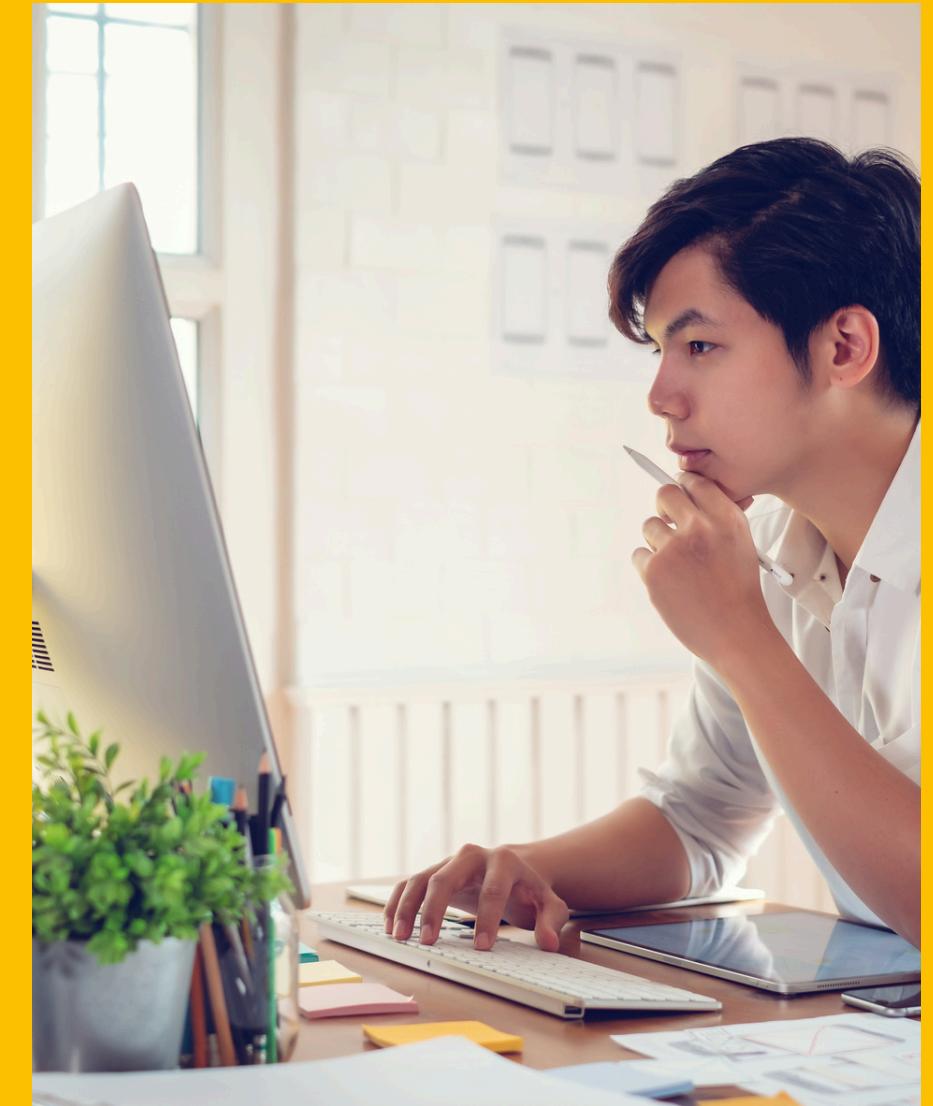
Handling Missing Value Pada Data Train Pada Fitur Age dan Fare

04

Handling Missing Value Pada Data Test Pada Fitur Age dan Fare

05

Mengambil Dataset Fitur Survived Yang Bukan Pos_missing_train



TOPIK OVERVIEW

06

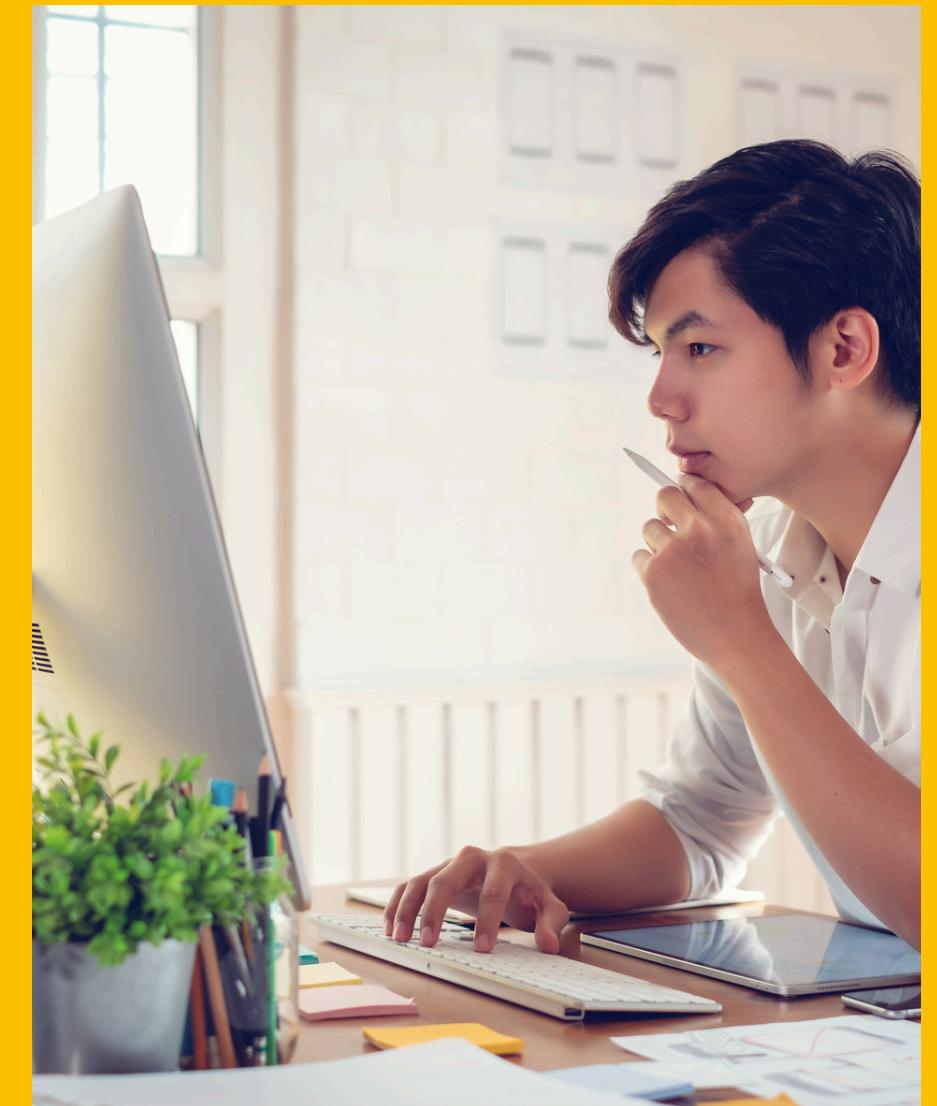
Menampilkan Test Label Yang
Bukan Pos_missing_test

07

Normalisasi MinMax Pada Data
Train

08

Normalisasi MinMax Pada Data
Test



TOPIK OVERVIEW

09

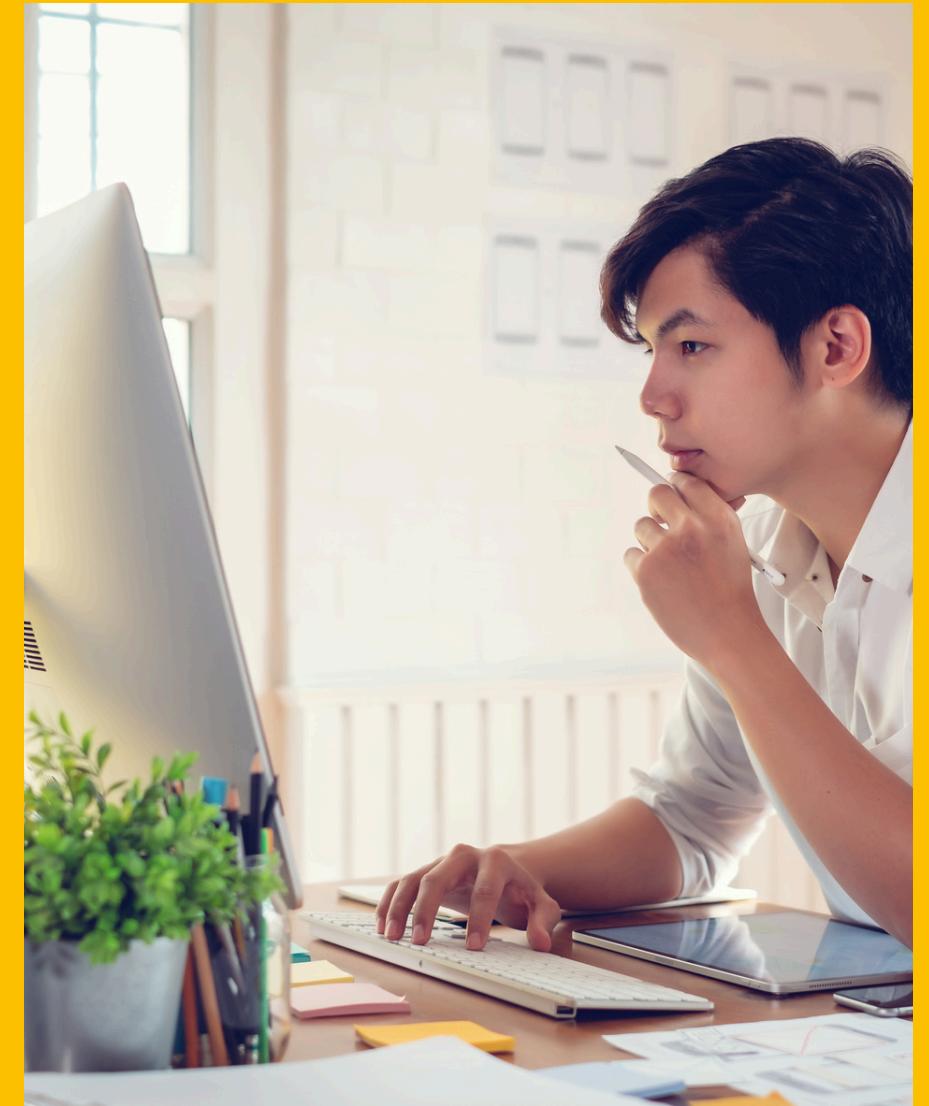
Klasifikasi KNN

10

Evaluasi Hasil Klasifikasi

11

Memilih Dan Menampilkan K terbaik



LATIHAN

CODING

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

# Membaca dataset Titanic
titanic_df = pd.read_csv("titanic.csv")

# Menghapus data yang kosong
titanic_df = titanic_df.dropna()

# Memilih fitur dan label
fitur = titanic_df[['Pclass', 'Age', 'SibSp', 'Parch', 'Fare']]
kelas = titanic_df[['Survived']]

# Membagi data menjadi training dan testing
training_data, testing_data, training_label, testing_label = train_test_split(
    fitur, kelas, test_size=0.2, random_state=42
)

# Membuat dan melatih model KNN
kNN = KNeighborsClassifier(n_neighbors=5, weights='distance')
kNN.fit(training_data, training_label)

# Memprediksi hasil
class_result = kNN.predict(testing_data)

# Menghitung precision ratio dan error ratio
precision_ratio = kNN.score(testing_data, testing_label)
error_ratio = 1 - precision_ratio

# Menampilkan error ratio
print(error_ratio)
```

OUTPUT

0.32432432432434

pertama meload dataset, setelah itu menghapus missing values. Selanjutnya membagi menjadi kolom dan label atau kelas target yakni fitu survived untuk dimasukkan ke dataframe yang baru. Setelah itu membagi data menjadi data tes,data train, data train label, dan data test label. Dengan data test sebesar 20%. Kemudian model di latih menggunakan algoritma KNN dengan k sebanyak 5, setelah itu prediksi terhadap data uji. Selanjutnya menghitung nilai error rasio dari hasil prediksi tadi. Didapatkan nilai error rasio sebesar 0.32432432432434 yang artinya 32.4% prediksi tidak sesuai dengan label asli

01. MEMBACA DATASET

```
df = pd.read_csv('titanic.csv')  
df
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th... Heikkinen, Miss. Laina	female	38.0	1	0	PC 17599 STON/O2. 3101282	71.2833 7.9250	C85 NaN	C S
2	3	1	3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	26.0	0	0	113803	53.1000	C123	S
3	4	1	1	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
4	5	0	3									
...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

Di tahap pertama ini saya meload dataset. Untuk klasifikasi kali ini saya akan menggunakan dataset titanic

02. MEMBAGI DATASET

DATA TEST

```
test = df.iloc[0:418]  
test
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	Cumings, Mrs. John Bradley (Florence Briggs Th... Heikkinen, Miss. Laina	female	38.0 26.0	1 0	0	PC 17599 STON/O2. 3101282	71.2833 7.9250	C85 NaN	C S
2	3	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
3	4	0	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
4	5	3									
...
413	414	0	Cunningham, Mr. Alfred Fleming	male	NaN	0	0	239853	0.0000	NaN	S
414	415	1	Sundman, Mr. Johan Julian	male	44.0	0	0	STON/O 2. 3101269	7.9250	NaN	S
415	416	0	Meek, Mrs. Thomas (Annie Louise Rowley)	female	NaN	0	0	343095	8.0500	NaN	S
416	417	1	Drew, Mrs. James Vivian (Lulu Thorne Christian)	female	34.0	1	1	28220	32.5000	NaN	S
417	418	1	Silven, Miss. Lyli Karoliina	female	18.0	0	2	250652	13.0000	NaN	S

418 rows × 12 columns

DATA TRAIN

```
train = df[['Age', 'Fare']]  
train.head(5)
```

✓ 0.0s	Age	Fare
0	22.0	7.2500
1	38.0	71.2833
2	26.0	7.9250
3	35.0	53.1000
4	35.0	8.0500

Selanjutnya membagi dataset menjadi 2 yakni mengambil 418 data pertama menjadi data test. Dan untuk data train saya menggunakan semua dataset.

04. HANDLING MISSING VALUE DATA TRAIN FITUR (AGE DAN FARE)

HANDLING MISSING VALUES

SEBELUM

```
train = df[['Age', 'Fare']]  
train
```

	Age	Fare
0	22.0	7.2500
1	38.0	71.2833
2	26.0	7.9250
3	35.0	53.1000
4	35.0	8.0500
...

886	27.0	13.0000
887	19.0	30.0000
888	NaN	23.4500
889	26.0	30.0000
890	32.0	7.7500

```
# Mencatat posisi missing value dalam dataset train sebelum di-drop  
pos_missing_train = pd.DataFrame()  
missing_indices_train = []  
  
for i in range(len(train)):  
    if train.iloc[i].isnull().any():  
        pos_missing_train = pd.concat([pos_missing_train, train.iloc[[i]]])  
        missing_indices_train.append(i)  
  
print("Posisi data yang memiliki missing value di train:", missing_indices_train)  
  
# Drop data yang memiliki missing value di train  
train = train.drop(pos_missing_train.index)  
  
# Mencatat posisi missing value pada fitur 'Age' dan 'Fare' di train  
pos_missing_age_test = np.where(test['Age'].isna())[0]  
pos_missing_fare_test = np.where(test['Fare'].isna())[0]
```

```
Posisi data yang memiliki missing value di train: [5, 17, 19, 26, 28, 29, 31, 32, 36, 42, 45, 46, 47, 48, 55, 64, 65, 76, 77, 82, 87, 95, 101, 107, 109, 121, 126, 128, 140, 154, 158,  
, 176, 180, 181, 185, 186, 196, 198, 201, 214, 223, 229, 235, 240, 241, 250, 256, 260, 264, 270, 274, 277, 284, 295, 298, 300, 301, 303, 304, 306, 324, 330, 351, 354, 358, 359, 364, 367, 368, 375, 384, 388, 409, 410, 411, 413, 415, 420, 425, 428, 431, 444, 451, 454, 457, 459, 464, 466, 468, 470, 475, 481, 485, 490, 495, 497, 502, 507, 511, 669, 674, 680, 692, 697, 709, 711, 718, 727, 732, 738, 739, 740, 760, 766, 768, 773, 776, 778, 783, 790, 792, 793, 815, 825, 826, 828, 832, 837, 839, 846, 849, 859, 863, 868, 878, 888]
```

SESUDAH

```
train = train[['Age', 'Fare']]  
train  
✓ 0.0s
```

	Age	Fare
0	22.0	7.2500
1	38.0	71.2833
2	26.0	7.9250
3	35.0	53.1000
4	35.0	8.0500
...
885	39.0	29.1250
886	27.0	13.0000
887	19.0	30.0000
888	26.0	30.0000
889	32.0	7.7500

Disini saya mengambil kolom age dan fare dari data train untuk dilakukan handling missing value dan menampilkan pada indeks keberapa missing value ini berada, nilai missing value ini akan disimpan ke dalam variable pos_missing_train. Setelah menampilkan letak missing value akan dilakukan penghapusan missing value. Dapat dilihat setelah penghapusan sudah tidak ditemukan nilai NaN

05. HANDLING MISSING VALUE DATA TES FITUR (AGE DAN FARE)

SEBELUM

```
test = test[['Age', 'Fare']]  
test  
✓ 0.0s
```

	Age	Fare
0	22.0	7.2500
1	38.0	71.2833
2	26.0	7.9250
3	35.0	53.1000
4	35.0	8.0500
...
413	NaN	0.0000
414	44.0	7.9250
415	NaN	8.0500
416	34.0	32.5000
417	18.0	13.0000

418 rows × 2 columns

HANDLING MISSING VALUES

```
# Mencatat posisi missing value dalam dataset test sebelum di-drop  
pos_missing_test = pd.DataFrame()  
missing_indices_test = []  
  
for i in range(len(test)):  
    if test.iloc[i].isnull().any():  
        pos_missing_test = pd.concat([pos_missing_test, test.iloc[[i]]])  
        missing_indices_test.append(i)  
  
print("Posisi data yang memiliki missing value di test:", missing_indices_test)  
  
# Drop data yang memiliki missing value di test  
test = test.drop(pos_missing_test.index)  
  
# Mencatat posisi missing value pada fitur 'Age' dan 'Fare' di tes|  
pos_missing_age = np.where(train['Age'].isna())[0]  
pos_missing_fare = np.where(train['Fare'].isna())[0]
```

SESUDAH

```
test = test[['Age', 'Fare']]  
test  
✓ 0.0s
```

	Age	Fare
0	22.0	7.2500
1	38.0	71.2833
2	26.0	7.9250
3	35.0	53.1000
4	35.0	8.0500
...
408	21.0	7.7750
412	33.0	90.0000
414	44.0	7.9250
416	34.0	32.5000
417	18.0	13.0000

335 rows × 2 columns

```
Posisi data yang memiliki missing value di test: [5, 17, 19, 26, 28, 29, 31, 32, 36, 42, 45, 46, 47, 48, 55, 64, 65, 76, 77, 82, 87, 95, 101, 107, 109, 121, 126, 128, 140, 154, 158,  
166, 168, 176, 180, 181, 185, 186, 196, 198, 201, 214, 223, 229, 235, 240, 241, 250, 256, 260, 264, 270, 274, 277, 284, 295, 298, 300, 301, 303, 304, 306, 324, 330, 334, 335, 347, 351, 354, 358, 359, 364, 367, 368, 375, 384, 388, 409, 410, 411, 413, 415]
```

Disini saya mengambil kolom age dan fare dari data tes untuk dilakukan handling missing value dan menampilkan pada indeks keberapa missing value ini berada, nilai missing value ini akan disimpan ke dalam variable pos_missing_test. Setelah menampilkan letak missing value akan dilakukan penghapusan missing value. Dapat dilihat setelah penghapusan sudah tidak ditemukan nilai NaN

06. AMBIL DATASET KOLOM KELAS (SURVIVED), YANG BUKAN POS_MISSING_TRAIN

CODING

```
df = df.drop(pos_missing_train.index)
train_label = df['survived']
train_label
```

Mengambil fitur survived, namun karena dataset awal masih terdapat missing value maka harus di handling missing values terlebih dahulu, di sini saya akan menghapus baris yang sama dengan variable pos_missing_values

OUTPUT

```
0      0
1      1
2      1
3      1
4      0
...
885    0
886    0
887    1
889    1
890    0
Name: Survived, Length: 714, dtype: int64
```

06. MENAMPAKILKAN TEST LABEL YANG BUKAN POS_MISSING_TEST

CODING

```
test_label = pd.read_csv ("titanic_testlabel.csv")
test_label = test_label.drop(pos_missing_test.index)
test_label = test_label.drop('PassengerId', axis=1)
test_label
```

Meload dataset baru yakni test label, selanjutnya menghapus baris yang ada missing value nya. Karena kolom survived tidak di pakai di sini saya menghapus kolom survived

OUTPUT

	Survived
0	0
1	1
2	0
3	0
4	1
...	...
408	1
412	1
414	1
416	0
417	0

335 rows × 1 columns

07. NORMALISASI DATA TRAIN DENGAN MIN MAX

TAHAP NORMALISASI

```
min_max_scaler = MinMaxScaler()
train_data_scaling = min_max_scaler.fit_transform(train)
print("Hasil scaling train data:")
print(train_data_scaling)

✓ 0.0s

Hasil scaling train data:
[[0.27117366 0.01415106]
 [0.4722292  0.13913574]
 [0.32143755 0.01546857]
 ...
 [0.23347575 0.0585561 ]
 [0.32143755 0.0585561 ]
 [0.39683338 0.01512699]]
```

NILAI MIN DAN MAX AGE DAN FARE

```
print('Nilai min dan max dari masing-masing fitur sebelum normalisasi (train):')
print("Nilai minimum:\n", train.min())
print("Nilai maksimum:\n", train.max())

print("\n" + "*"*50)
print('Nilai min dan max dari masing-masing fitur setelah normalisasi (train):')
print("Nilai minimum:\n", train_data_scaling.min())
print("Nilai maksimum:\n", train_data_scaling.max())
✓ 0.0s

Nilai min dan max dari masing-masing fitur sebelum normalisasi (train):
Nilai minimum:
Age    0.42
Fare   0.00
dtype: float64
Nilai maksimum:
Age    80.0000
Fare   512.3292
dtype: float64

=====
Nilai min dan max dari masing-masing fitur setelah normalisasi (train):
Nilai minimum:
0.0
Nilai maksimum:
1.0
```

Melakukan standarisasi dengan menggunakan metode min-max pada data train. Sebelum dilakukan normalisasi nilai min dari fitur age yakni 0.42 dan nilai max sebesar 80.0000 sedangkan nilai min pada fitur fare 0 dan nilai max sebesar 512.3292. nilai ini menunjukkan fitur age dan fare memiliki skala nilai yang sangat jauh sehingga harus dilakukan normalisasi. Setelah normalisasi nilai min pada fitur age dan fare sebesar 0 dan nilai max pada kedua fitur ini adalah 1

08. NORMALISASI DATA TEST DENGAN MIN MAX

TAHAP NORMALISASI

```
test_data_scaling = min_max_scaler.transform(test)
print("\nHasil scaling test data:")
print(test_data_scaling)

✓ 0.0s
```

Hasil scaling test data:

```
[[0.27117366 0.01415106]
 [0.4722292 0.13913574]
 [0.32143755 0.01546857]
 [0.43453129 0.1036443 ]
 [0.43453129 0.01571255]
 [0.67328474 0.10122886]
 [0.01985423 0.04113566]
 [0.33400352 0.02173075]
 [0.17064589 0.05869429]
 [0.04498618 0.03259623]
 [0.72354863 0.05182215]
 [0.24604172 0.01571255]
 [0.48479517 0.06104473]]
```

NILAI MIN DAN MAX AGE DAN FARE

```
print('Nilai min dan max dari masing-masing fitur sebelum normalisasi (test):')
print("Nilai minimum:\n", test.min())
print("Nilai maksimum:\n", test.max())

print("\n" + "*50)
print('Nilai min dan max dari masing-masing fitur setelah normalisasi (test):')
print("Nilai minimum:\n", test_data_scaling.min())
print("Nilai maksimum:\n", test_data_scaling.max())

✓ 0.0s

Nilai min dan max dari masing-masing fitur sebelum normalisasi (test):
Nilai minimum:
Age      0.83
Fare     0.00
dtype: float64
Nilai maksimum:
Age      71.0000
Fare    512.3292
dtype: float64

=====
Nilai min dan max dari masing-masing fitur setelah normalisasi (test):
Nilai minimum:
0.0
Nilai maksimum:
1.0
```

Melakukan standarisasi dengan menggunakan metode min-max pada data tes. Sebelum dilakukan normalisasi nilai min dari fitur age yakni 0.83 dan nilai max sebesar 71.0000 sedangkan nilai min pada fitur fare 0 dan nilai max sebesar 512.3292. Nilai ini menunjukkan fitur age dan fare memiliki skala nilai yang sangat jauh sehingga harus dilakukan normalisasi. Setelah normalisasi, nilai min pada fitur age dan fare sebesar 0 dan nilai max pada kedua fitur ini adalah 1

09. KLASIFIKASI DENGAN KNN

CODING

```
class_results = {}
results_list = []

for k in range(1, 11):
    kNN = KNeighborsClassifier(n_neighbors=k)
    kNN.fit(train_data_scaling, train_label)
    class_result = kNN.predict(test_data_scaling)

    # Simpan hasil prediksi untuk analisis selanjutnya
    class_results[k] = class_result

    # Menghitung error ratio dan akurasi
    precision_ratio = kNN.score(test_data_scaling, test_label)
    error_ratio = 1 - precision_ratio

    # Simpan ke dalam list untuk DataFrame
    results_list.append({'k': k, 'Error Ratio': error_ratio, 'Accuracy': precision_ratio})

# Membuat DataFrame dari hasil evaluasi
results_df = pd.DataFrame(results_list)
# Menampilkan hasil evaluasi dalam bentuk tabel
print("\nHasil Evaluasi KNN:")
print(results_df)
```

OUTPUT

Hasil Evaluasi KNN:			
k	Error Ratio	Accuracy	
0	0.510448	0.489552	
1	0.438806	0.561194	
2	0.516418	0.483582	
3	0.429851	0.570149	
4	0.465672	0.534328	
5	0.450746	0.549254	
6	0.486567	0.513433	
7	0.480597	0.519403	
8	0.504478	0.495522	
9	0.483582	0.516418	

selanjutnya di sini saya melakukan klasifikasi pada data tes dan data train yang telah dilakukan normalisasi dengan beberapa iterasi dengan k sebanyak 10. Pada setiap iterasi model dilatih dan diuji dengan k=1 hingga k=10

10. MENAMPILKAN K TERBAIK

CODING

```
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report
# Mencari k terbaik berdasarkan jumlah kesalahan prediksi
best_k = min(class_results.keys(), key=lambda k: sum(class_results[k] != test_label['Survived'].values))
best_error_ratio = sum(class_results[best_k] != test_label['Survived'].values) / len(test_label)

print("\nNilai k terbaik:", best_k)
print(f"Error ratio terbaik: {best_error_ratio:.4f}")
print(f"Accuracy terbaik: {1 - best_error_ratio:.4f}")

# Menampilkan classification report untuk k terbaik
print("\nClassification Report untuk k terbaik:")
print(classification_report(test_label['Survived'], class_results[best_k]))
```

OUTPUT

```
Nilai k terbaik: 4
Error ratio terbaik: 0.4299
Accuracy terbaik: 0.5701

Classification Report untuk k terbaik:
precision    recall   f1-score   support
          0       0.64      0.75      0.69      214
          1       0.36      0.25      0.29      121
accuracy                           0.57      335
macro avg       0.50      0.50      0.49      335
weighted avg    0.54      0.57      0.55      335
```

Disini saya juga membuat program agar bisa menampilkan k yang terbaik yang memiliki error ratio terendah dan akurasi tertinggi. Didapatkan dari klasifikasi dengan knn, k terbaik berada pada k=4 dengan error ratio terendah yakni .4299 dan accuracy terbesar yakni 0.5701



LINK COLAB

[LINK KODING KLASIFIKASI](#)

11

TERIMA KASIH

