Tahapan validasi metode klasifikasi

1. Tentukan metode klasifikasi yang akan dievaluasi

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC


#Spot Check Algorithm
models = []
models.append(('KNN', KNeighborsClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC(gamma='auto')))

# evaluate each model in turn
results = []
names = []
```

2. Baca dataset

```
! wget https://archive.ics.uci.edu/ml/machine-learning-databases/00451/dataR2.csv
```

```
--2020-12-03 02:27:30--  https://archive.ics.uci.edu/ml/machine-learning-databases/00451/dataR2.csv
Resolving archive.ics.uci.edu (archive.ics.uci.edu)... 128.195.10.252
Connecting to archive.ics.uci.edu (archive.ics.uci.edu)|128.195.10.252|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 7665 (7.5K) [application/x-httpd-php]
Saving to: ‘dataR2.csv’

dataR2.csv          100%[===================>]   7.49K  --.-KB/s    in 0s

2020-12-03 02:27:31 (135 MB/s) - ‘dataR2.csv’ saved [7665/7665]
```

3. Tampilkan dataset

```
import pandas as pd
import numpy as np

dataset = pd.read_csv('dataR2.csv')
dataset
```

	Age	BMI	Glucose	Insulin	HOMA	Leptin	Adiponectin	Resistin	MCP.1	Classification
0	48	23.500000	70	2.707	0.467409	8.8071	9.702400	7.99585	417.114	1
1	83	20.690495	92	3.115	0.706897	8.8438	5.429285	4.06405	468.786	1
2	82	23.124670	91	4.498	1.009651	17.9393	22.432040	9.27715	554.697	1
3	68	21.367521	77	3.226	0.612725	9.8827	7.169560	12.76600	928.220	1
4	86	21.111111	92	3.549	0.805386	6.6994	4.819240	10.57635	773.920	1
...	...	...	...	...	...	...	...	...	...	...
111	45	26.850000	92	3.330	0.755688	54.6800	12.100000	10.96000	268.230	2
112	62	26.840000	100	4.530	1.117400	12.4500	21.420000	7.32000	330.160	2
113	65	32.050000	97	5.730	1.370998	61.4800	22.540000	10.33000	314.050	2
114	72	25.590000	82	2.820	0.570392	24.9600	33.750000	3.27000	392.460	2
115	86	27.180000	138	19.910	6.777364	90.2800	14.110000	4.35000	90.090	2

116 rows × 10 columns

```
dataset.head()
```

	Age	BMI	Glucose	Insulin	HOMA	Leptin	Adiponectin	Resistin	MCP.1	Classification
0	48	23.500000	70	2.707	0.467409	8.8071	9.702400	7.99585	417.114	1
1	83	20.690495	92	3.115	0.706897	8.8438	5.429285	4.06405	468.786	1
2	82	23.124670	91	4.498	1.009651	17.9393	22.432040	9.27715	554.697	1

3.1 Membuat Set Data Validasi

```
from sklearn.model_selection import train_test_split
array = dataset.values
X = array[:,0:9]
y = array[:,9]
X_train, X_validation, Y_train, Y_validation = train_test_split(X, y, test_size=0.20, random_state=1, shuffle=True)
```

3.2 K-Fold Cross Validation

```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold

for name, model in models:
  kfold = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)
  cv_results = cross_val_score(model, X_train, Y_train, cv=kfold, scoring='accuracy')
  results.append(cv_results)
  names.append(name)

  print('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))
```

```
KNN: 0.487778 (0.143540)
NB: 0.662222 (0.125629)
SVM: 0.465556 (0.047778)
```

4. Buat Prediksi

4.1 Membuat Prediksi

```
model = SVC(gamma='auto')
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
```

4.2 Evaluasi Prediksi

```
# Create Dataset
from sklearn.datasets import make_classification
x, y = make_classification(n_samples=1000, n_features=20, n_informative=15, n_redundant=5, random_state=1)

print(x)
print(y)
```

```
     [[  2.47475454   0.40165523   1.68081787 ...  -6.59044146  -2.21290585
        -3.139579  ]
      [  0.84802507   2.81841945  -2.76008732 ...   3.00844461   0.78661954
        -1.27681551]
      [ -1.90041246  -0.56901823  -1.76220236 ...   3.37336417  -2.28613707
         1.90344983]
      ...
      [  0.7673844   -2.91920559   2.80851577 ...   4.42591832   0.46321196
        -3.30523346]
      [  2.05510667  -0.99009741   0.73577291 ...   3.05100898  -1.40715279
        -0.51579331]
      [-10.96847792  -2.39810735  -0.96700953 ... -11.16298557   1.16646392
         0.60835176]]
     [0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 0 1 0 1 0 1 1 1 1 1 1 1 1 1 1 0 0
      0 1 0 0 0 1 0 0 1 1 1 0 0 0 1 1 1 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 1 0 1 1 1
      1 1 1 1 1 1 1 0 1 0 1 1 1 0 1 1 0 0 1 0 0 1 1 1 0 0 1 1 1 1 1 1 0 1 0 0 1
      1 0 1 0 0 1 1 1 0 1 0 0 1 1 1 0 1 0 1 0 0 1 0 1 0 0 0 0 1 0 1 1 0 0 1 0 1
      0 0 0 1 1 0 1 1 0 0 0 0 1 0 0 0 0 1 0 1 0 1 0 0 1 0 1 1 0 0 0 1 0 1 1 0 0
      0 0 1 1 1 1 0 0 1 0 0 0 1 1 0 1 1 0 1 0 0 1 0 1 0 0 0 1 0 0 1 1 1 0 0 0 1
      0 1 1 0 1 1 0 1 0 1 0 0 0 0 0 0 1 1 1 1 0 0 1 1 0 1 1 1 0 0 1 0 0 1 0 1 0
      1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 0 1 0 0 1 1 0 0 0 1 1 0 0 1 0 0 1 1 1
      0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 1 1 1 0 0 1 0 1 0 1 0 0 0 0 1 1 1 1 0
```

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

```
from sklearn.metrics import classification_report
print(classification_report(Y_validation, predictions))
```

```
                precision    recall  f1-score   support

           1.0       0.25      1.00      0.40         6
           2.0       0.00      0.00      0.00        18

      accuracy                           0.25        24
     macro avg       0.12      0.50      0.20        24
  weighted avg       0.06      0.25      0.10        24

/usr/local/lib/python3.6/dist-packages/sklearn/metrics/_classification.py:1272: UndefinedMetricWarning: Precision and F
  _warn_prf(average, modifier, msg_start, len(result))
```

Tugas

1. Lakukan perbandingan klasifikasi antara SVM, Naive Bayes, KNN untuk dataset iris dan breast cancer

1.1 Dataset Iris

1.1.1 Menyiapkan dataset iris

```
! wget https://dataset-ppm.s3.amazonaws.com/iris.csv
```

```
--2020-12-03 02:28:08--  https://dataset-ppm.s3.amazonaws.com/iris.csv
Resolving dataset-ppm.s3.amazonaws.com (dataset-ppm.s3.amazonaws.com)... 52.216.240.20
Connecting to dataset-ppm.s3.amazonaws.com (dataset-ppm.s3.amazonaws.com)|52.216.240.20|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4609 (4.5K) [text/csv]
Saving to: ‘iris.csv’

iris.csv            100%[===================>]   4.50K  --.-KB/s    in 0s

2020-12-03 02:28:08 (162 MB/s) - ‘iris.csv’ saved [4609/4609]
```

```
dataset_iris = pd.read_csv('iris.csv')
dataset_iris
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa

```
dataset_iris.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

1.1.2 Menyiapkan Set Validasi Data Iris

```
array = dataset_iris.values
X = array[:,0:4]
y = array[:,4]
X_train, X_validation, Y_train, Y_validation = train_test_split(X, y, test_size=0.20, random_state=1, shuffle=True)
```

1.1.3 K-Fold Cross Validation Data Iris

```
results_iris = []
names_iris = []

for name, model in models:
  kfold = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)
  iris_results = cross_val_score(model, X_train, Y_train, cv=kfold, scoring='accuracy')
  results.append(iris_results)
  names.append(name)
```

```
print('%s: %f (%f)' % (name, iris_results.mean(), iris_results.std()))
```

```
KNN: 0.958333 (0.041667)
NB: 0.950000 (0.055277)
SVM: 0.983333 (0.033333)
```

1.2 Dataset Breast Cancer

1.2.1 Menyiapkan dataset breast cancer (bc)

```
! wget https://raw.githubusercontent.com/frnkldgnwn/utp-ppm/main/breast_cancer.csv
```

```
--2020-12-03 02:28:25--  https://raw.githubusercontent.com/frnkldgnwn/utp-ppm/main/breast_cancer.csv
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.0.133, 151.101.64.133, 151.101.128.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.0.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3664 (3.6K) [text/plain]
Saving to: ‘breast_cancer.csv’

breast_cancer.csv   100%[===================>]   3.58K  --.-KB/s    in 0s

2020-12-03 02:28:25 (53.5 MB/s) - ‘breast_cancer.csv’ saved [3664/3664]
```

```
#karena dataset_bc berisi index, maka dataset_bc di-read ulang dengan index
dataset_bc = pd.read_csv('breast_cancer.csv', index_col=0)
dataset_bc
```

id	clump_thickness	cell_size	cell_shape	marginal_adhesion	epithelial_cell_size	bare_nucklei	bland_chromati
1000025	5.0	1.0	NaN	1.0	NaN	1.0	3.(
1002945	5.0	4.0	4.0	5.0	7.0	NaN	3.(
1015425	3.0	1.0	1.0	1.0	NaN	2.0	3.(
1016277	6.0	8.0	8.0	1.0	3.0	NaN	3.(
1017023	4.0	1.0	1.0	3.0	NaN	1.0	3.(
...	...	...	...	...	...	...	..
1167439	2.0	3.0	4.0	4.0	2.0	5.0	2.(

```
dataset_bc.head()
```

id	clump_thickness	cell_size	cell_shape	marginal_adhesion	epithelial_cell_size	bare_nucklei	bland_chromati
1000025	5.0	1.0	NaN	1.0	NaN	1.0	3.(
1002945	5.0	4.0	4.0	5.0	7.0	NaN	3.(
1015425	3.0	1.0	1.0	1.0	NaN	2.0	3.(
1016277	6.0	8.0	8.0	1.0	3.0	NaN	3.(
1017023	4.0	1.0	1.0	3.0	NaN	1.0	3.(

1.2.2 Menyiapkan Set Validasi Data Breast Cancer

```
array_bc = dataset_bc.values
X_bc = array_bc[:,0:9]
y_bc = array_bc[:,9]
X_train_bc, X_validation_bc, Y_train_bc, Y_validation_bc = train_test_split(X_bc, y_bc, test_size=0.20, random_state=1, shuf
```

1.2.3 K-Fold Cross Validation Data Breast Cancer

```
results_bc = []
names_bc = []

for name, model in models:
 kfold = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)
 bc_results = cross_val_score(model, X_train_bc, Y_train_bc, cv=kfold, scoring='accuracy')
 results_bc.append(bc_results)
 names_bc.append(name)
 print('%s: %f (%f)' % (name, bc_results.mean(), bc_results.std()))
```

```
ValueError: Input contains NaN, infinity or a value too large for dtype('float64').

  FitFailedWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_validation.py:536: FitFailedWarning: Estimator fit fa
ValueError: Input contains NaN, infinity or a value too large for dtype('float64').

  FitFailedWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_validation.py:536: FitFailedWarning: Estimator fit fa
ValueError: Input contains NaN, infinity or a value too large for dtype('float64').

  FitFailedWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_validation.py:536: FitFailedWarning: Estimator fit fa
ValueError: Input contains NaN, infinity or a value too large for dtype('float64').

  FitFailedWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_validation.py:536: FitFailedWarning: Estimator fit fa
ValueError: Input contains NaN, infinity or a value too large for dtype('float64').

  FitFailedWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_validation.py:536: FitFailedWarning: Estimator fit fa
ValueError: Input contains NaN, infinity or a value too large for dtype('float64').

  FitFailedWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_validation.py:536: FitFailedWarning: Estimator fit fa
ValueError: Input contains NaN, infinity or a value too large for dtype('float64').

  FitFailedWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_validation.py:536: FitFailedWarning: Estimator fit fa
ValueError: Input contains NaN, infinity or a value too large for dtype('float64').
```

```
  FitFailedWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_validation.py:536: FitFailedWarning: Estimator fit fa
ValueError: Input contains NaN, infinity or a value too large for dtype('float64').

  FitFailedWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_validation.py:536: FitFailedWarning: Estimator fit fa
ValueError: Input contains NaN, infinity or a value too large for dtype('float64').

  FitFailedWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_validation.py:536: FitFailedWarning: Estimator fit fa
ValueError: Input contains NaN, infinity or a value too large for dtype('float64').

  FitFailedWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_validation.py:536: FitFailedWarning: Estimator fit fa
ValueError: Input contains NaN, infinity or a value too large for dtype('float64').

  FitFailedWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_validation.py:536: FitFailedWarning: Estimator fit fa
ValueError: Input contains NaN, infinity or a value too large for dtype('float64').

  FitFailedWarning)

/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_validation.py:536: FitFailedWarning: Estimator fit fa
ValueError: Input contains NaN, infinity or a value too large for dtype('float64').

  FitFailedWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_validation.py:536: FitFailedWarning: Estimator fit fa
ValueError: Input contains NaN, infinity or a value too large for dtype('float64').
```

Dari hasil di atas terlihat bahwa K-Fold Cross Validation tidak menerima data yang mengandung NaN. Maka perlu dilakukan preproses dan normalisasi data terlebih dahulu jika ingin melihat hasilnya.

2. Pada kedua dataset lakukan proses normalisasi data dan preproses data untuk menengani data yang hilang jika ada.

2.1 Dataset iris

2.1.1 Preproses Dataset Iris

```
dataset_irisPre = dataset_iris.replace(0.0, np.NaN)


def imputasi(df_input):
  list_columns = df_input.columns
  class_column = list_columns[-1]
  for column in list_columns[:-1]:
    df_input[column] = df_input[column].fillna(df_input.groupby(class_column)[column].transform('mean'))
  return df_input


dataset_irisImp = imputasi(dataset_irisPre)
```

2.1.2 Normalisasi Min-Max Dataset Iris

```
def minmax(df_input):
  list_fitur = df_input.columns[:-1]
  for fitur in list_fitur:
    max = df_input[fitur].max()
    min = df_input[fitur].min()
    df_input[fitur] = (df_input[fitur]-min)/(max-min)
  return df_input


dataset_irisN = minmax(dataset_irisImp)


dataset_irisN.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	0.222222	0.625000	0.067797	0.041667	Iris-setosa
2	0.111111	0.500000	0.050847	0.041667	Iris-setosa

2.1.3 Set Validasi Data Iris Normal

```
array_irisN = dataset_irisN.values
X_irN = array_irisN[:,0:4]
y_irN = array_irisN[:,4]
X_train_irN, X_validation_irN, Y_train_irN, Y_validation_irN = train_test_split(X_irN, y_irN, test_size=0.20, random_state=1
```

2.1.4 K-Fold Cross Validation Data Iris Normal

```
results_irN = []
names_irN = []
for name, model in models:
 kfold = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)
 irisN_results = cross_val_score(model, X_train_irN, Y_train_irN, cv=kfold, scoring='accuracy')
 results_irN.append(irisN_results)
 names_irN.append(name)
 print('%s: %f (%f)' % (name, irisN_results.mean(), irisN_results.std()))
```

```
KNN: 0.950000 (0.040825)
NB: 0.950000 (0.055277)
SVM: 0.941667 (0.053359)
```

2.2 Dataset Breast Cancer

2.2.1 Preproses Dataset Breast Cancer

```
dataset_bcPre = dataset_bc.replace(0.0, np.NaN)
```

```
dataset_bcImp = imputasi(dataset_bcPre)
```

2.2.2 Normalisasi Min-Max Dataset Breast Cancer

```
dataset_bcN = minmax(dataset_bcImp)
```

```
dataset_bcN.head()
```

id	clump_thickness	cell_size	cell_shape	marginal_adhesion	epithelial_cell_size	bare_nucklei	bland_chromati
1000025	0.444444	0.000000	0.070707	0.000000	0.128019	0.000000	0.2
1002945	0.444444	0.333333	0.333333	0.444444	0.666667	0.041344	0.2
1015425	0.222222	0.000000	0.000000	0.000000	0.128019	0.111111	0.2
1016277	0.555556	0.777778	0.777778	0.000000	0.222222	0.041344	0.2
1017023	0.333333	0.000000	0.000000	0.222222	0.128019	0.000000	0.2

2.2.3 Set Validasi Data Breast Cancer Normal

```
array_bcN = dataset_bcN.values
X_bcN = array_bcN[:,0:9]
y_bcN = array_bcN[:,9]
X_train_bcN, X_validation_bcN, Y_train_bcN, Y_validation_bcN = train_test_split(X_bcN, y_bcN, test_size=0.20, random_state=1
```

2.2.4 K-Fold Cross Validation Data Breast Cancer Normal

```
results_bcN = []
names_bcN = []

for name, model in models:
 kfold = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)
 bcN_results = cross_val_score(model, X_train_bcN, Y_train_bcN, cv=kfold, scoring='accuracy')
```

```
    results_bcN.append(bcN_results)
    names_bcN.append(name)
    print('%s: %f (%f)' % (name, bcN_results.mean(), bcN_results.std()))
```

```
    KNN: 0.925000 (0.100000)
    NB: 0.900000 (0.108972)
    SVM: 0.937500 (0.083853)
```