Tupro3

by Group
LukaMagic

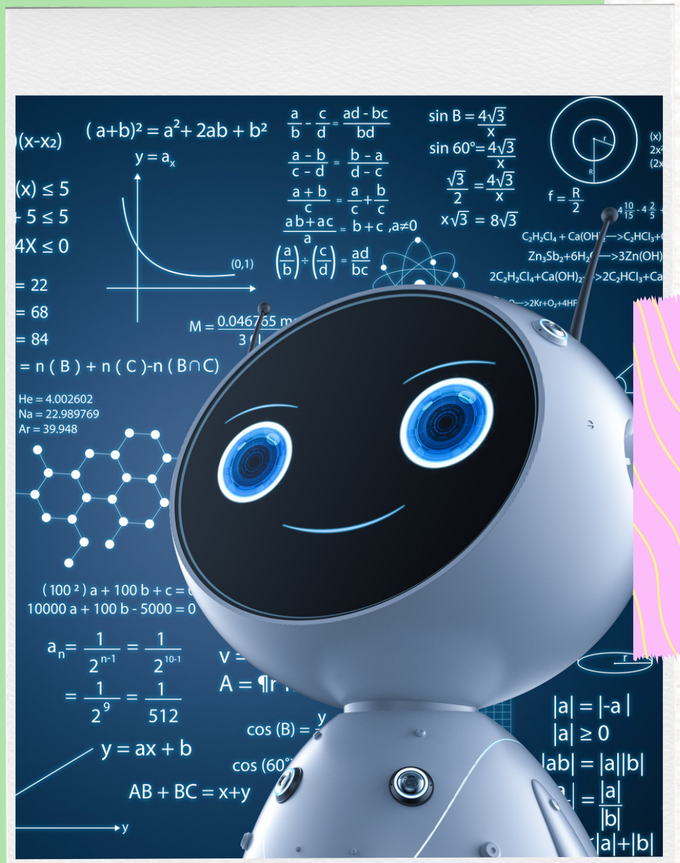
Team

Fadil

1301204297

Dafer

1301200076



Description

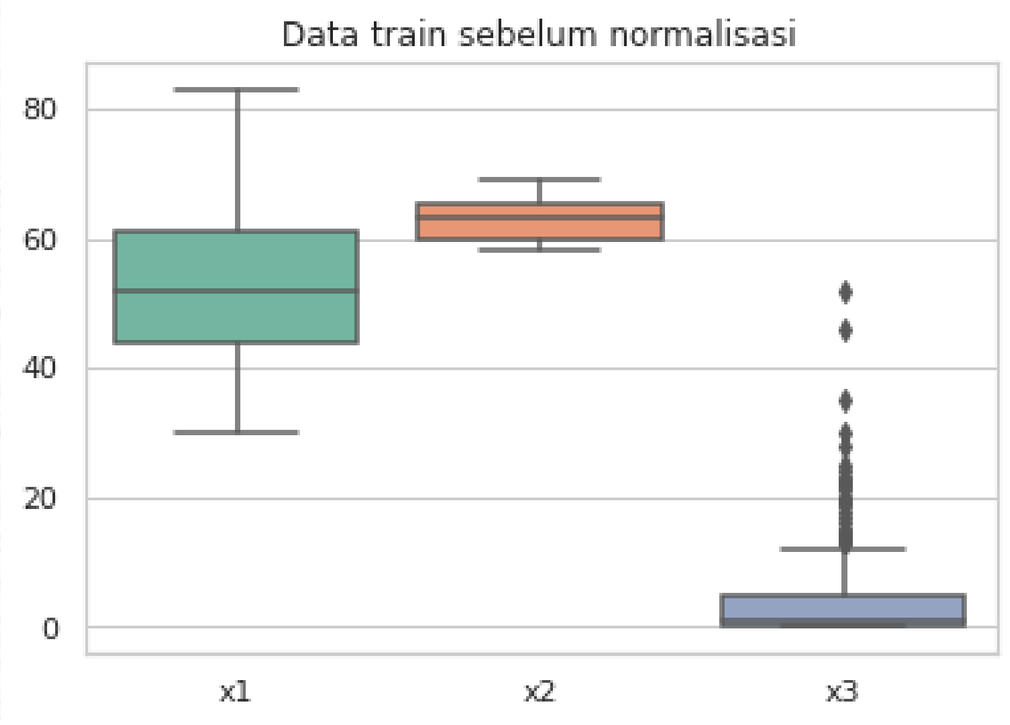
Diberikan file traintest.xlsx yang terdiri dari dua sheet: train dan test

Gunakan sheet ini untuk tahap pengujian (testing) model yang sudah dilatih.

Import data & Normalization

data train sebelum normalisasi

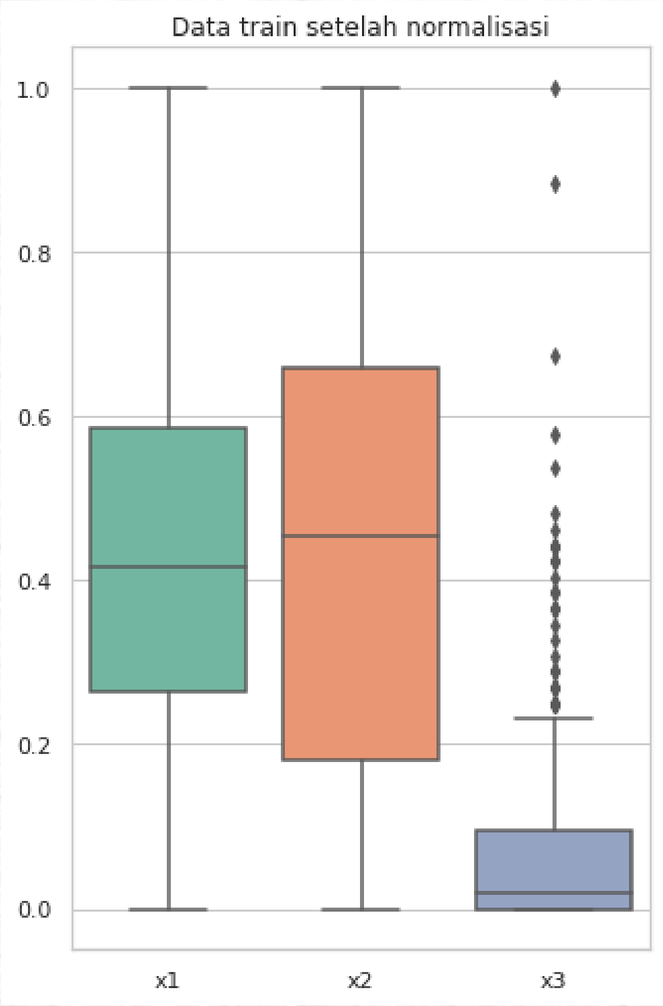
	id	x1	x2	x3	y
0	1	60	64	0	1
1	2	54	60	11	0
2	3	65	62	22	0
3	4	34	60	0	1
4	5	38	69	21	0
...	...	...	...	...	...
291	292	59	64	1	1
292	293	65	67	0	1
293	294	53	65	12	0
294	295	57	64	1	0
295	296	54	59	7	1



Import data & Normalization

data train sesudah normalisasi

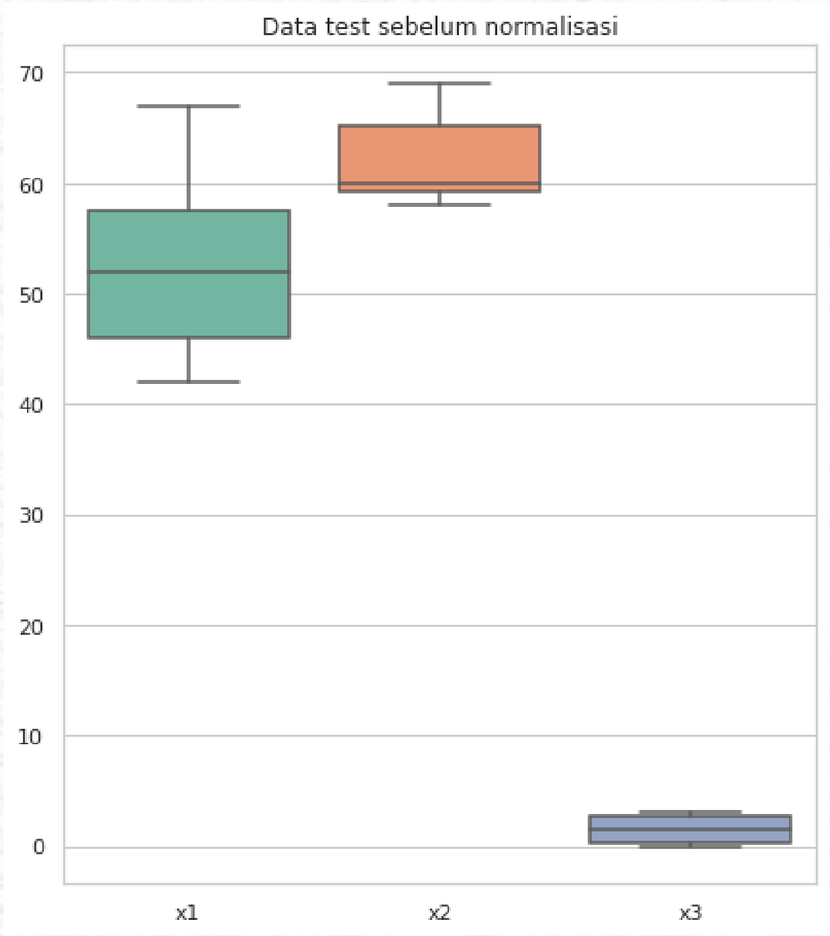
	id	x1	x2	x3	y
0	1	0.566038	0.545455	0.000000	1
1	2	0.452830	0.181818	0.211538	0
2	3	0.660377	0.363636	0.423077	0
3	4	0.075472	0.181818	0.000000	1
4	5	0.150943	1.000000	0.403846	0
...	...	...	...	...	...
291	292	0.547170	0.545455	0.019231	1
292	293	0.660377	0.818182	0.000000	1
293	294	0.433962	0.636364	0.230769	0
294	295	0.509434	0.545455	0.019231	0
295	296	0.452830	0.090909	0.134615	1



Import data & Normalization

data test sebelum normalisasi

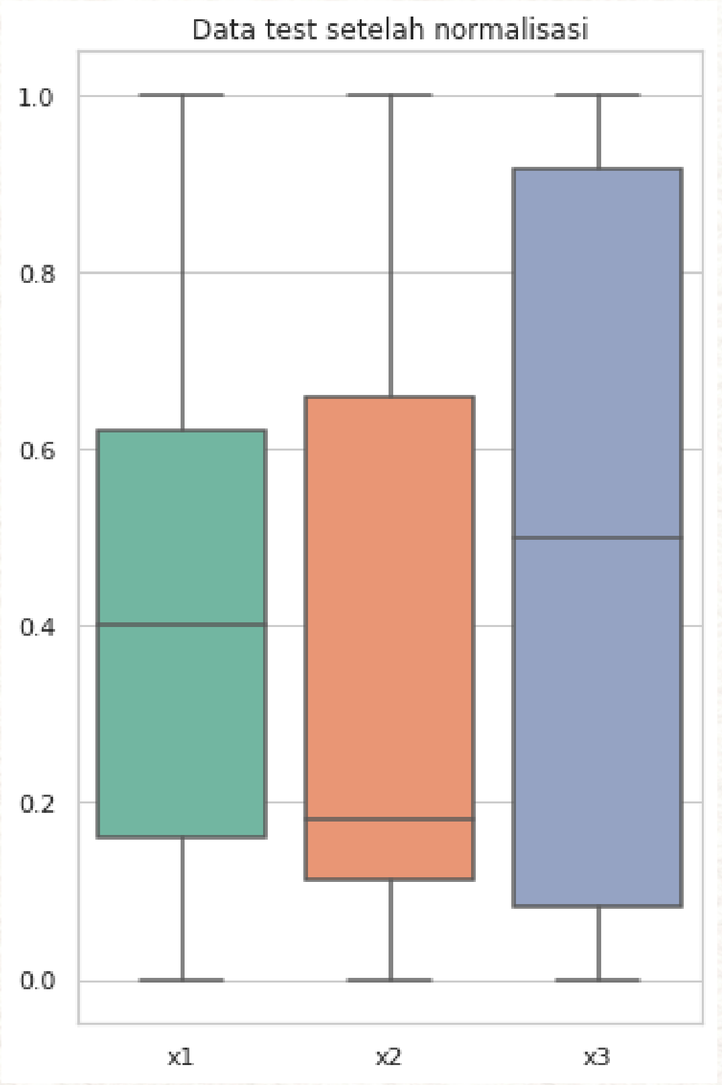
	id	x1	x2	x3
0	297	43	59	2
1	298	67	66	0
2	299	58	60	3
3	300	49	63	3
4	301	45	60	0
5	302	54	58	1
6	303	56	66	3
7	304	42	69	1
8	305	50	59	2
9	306	59	60	0



Import data & Normalization

data test sesudah normalisasi

	id	x1	x2	x3
0	297	0.04	0.090909	0.666667
1	298	1.00	0.727273	0.000000
2	299	0.64	0.181818	1.000000
3	300	0.28	0.454545	1.000000
4	301	0.12	0.181818	0.000000
5	302	0.48	0.000000	0.333333
6	303	0.56	0.727273	1.000000
7	304	0.00	1.000000	0.333333
8	305	0.32	0.090909	0.666667
9	306	0.68	0.181818	0.000000



Method

K-Nearest Neighbors

Naïve Bayes

$$P(A) = \frac{n(A)}{n(S)}$$

KNN

algoritma klasifikasi yang bekerja dengan mengambil sejumlah k data terdekat sebagai acuan menentukan kelas dari data yang baru.

KNN

Euclidean

$$d_1(x1, x2) = \sqrt{\sum_p (x1_p - x2_p)^2}$$

Manhattan

$$d_1(x1, x2) = \sum_p |x1_p - x2_p|$$

Euclidean

```python
def Euclidean(dfNorm, test):
    result = []
    for i in range(len(dfNorm)):
        distancex = math.sqrt((((dfNorm['x1'][i] - test['x1']) ** 2) +
                               ((dfNorm['x2'][i] - test['x2']) ** 2) +
                               ((dfNorm['x3'][i] - test['x3']) ** 2)))
        result.append([distancex, dfNorm['y'][i]])
    return result
```

```python
def KNNeuclidean(dfNorm, dfTestNorm, k):
    hasil = []
    for i in range(len(dfTestNorm)):
        distance = Euclidean(dfNorm, dfTestNorm.iloc[[i]])
        distance = sorted(distance, key=lambda x:x[0])
        distanceK = distance[:k]
        satu = 0
        nol = 0
        for j in range(k):
            if distanceK[j][1] == 1:
                satu += 1
            else:
                nol += 1
        if satu > nol:
            hasil.append([dfTestNorm.loc[i, 'id'], 1])
        else:
            hasil.append([dfTestNorm.loc[i, 'id'], 0])
    dfHasil = pd.DataFrame(hasil, columns = ['id', 'y'])
    return dfHasil
```

Validation

k Fold Validation ; k = 5

Fold 1 = dfNorm[:59]

Fold 2 = dfNorm[59:118]

Fold 3 = dfNorm[118:177]

Fold 4 = dfNorm[177:236]

Fold 5 = dfNorm[236:]

Validation

KNN Validation

K\F	1	2	3	4	5
3	66%	80%	78%	54%	71%
7	67%	81%	78%	54%	71%
11	70%	83%	81%	56%	74%
15	70%	84%	84%	61%	74%
17	71%	83%	83%	54%	74%
Average	69%	82%	81%	56%	73%

Euclidean

K = 15

id	y
297	0
298	1
299	0
300	0
301	1
302	0
303	0
304	1
305	0
306	1

Manhattan

```
def Manhattan(dfNorm, test):
    result = []
    for i in range(len(dfNorm)):
        distance = (abs(dfNorm['x1'][i] - test['x1']) +
                    abs(dfNorm['x2'][i] - test['x2']) +
                    abs(dfNorm['x3'][i] - test['x3']))
        result.append([distance , dfNorm['y'][i]])
    return result
```

Manhattan

K = 15

id	y
297	1
298	1
299	0
300	0
301	1
302	0
303	1
304	1
305	0
306	1

Naïve Bayes

algoritma yang mempelajari probabilitas suatu objek dengan ciri-ciri tertentu yang termasuk dalam kelompok/kelas tertentu.

Naïve Bayes

Gaussian Model

$$\hat{P}(X_j|C = c_i) = \frac{1}{\sigma_{ji}\sqrt{2\pi}} \exp\left(-\frac{(X_j - \mu_{ji})^2}{2\sigma_{ji}^2}\right)$$

Naïve Bayes

```python
def pisah(dfNorm):
  nol = []
  satu = []
  for i in range(len(dfNorm)):
      if dfNorm['y'][i] == 0:
          nol.append([dfNorm['x1'][i], dfNorm['x2'][i], dfNorm['x3'][i], dfNorm['y'][i]])
      else:
          satu.append([dfNorm['x1'][i], dfNorm['x2'][i], dfNorm['x3'][i], dfNorm['y'][i]])
  return nol, satu
```

```python
def getMeanVar(dfNorm):
    nol, satu = pisah(dfNorm)

    dfNol = pd.DataFrame(data = nol, columns = ['x1', 'x2', 'x3', 'y'])
    dfSatu = pd.DataFrame(data = satu, columns = ['x1', 'x2', 'x3', 'y'])

    nolMean = dfNol[['x1', 'x2', 'x3']].mean()
    satuMean = dfSatu[['x1', 'x2', 'x3']].mean()

    nolVar = dfNol[['x1', 'x2', 'x3']].var()
    satuVar = dfSatu[['x1', 'x2', 'x3']].var()

    return dfNol, dfSatu, nolMean, satuMean, nolVar, satuVar
```

Naïve Bayes

```
def Bayes(dfNorm, dfTestNorm):
    hasilBayes = []
    dfNol, dfSatu, nolMean, satuMean, nolVar, satuVar = getMeanVar(dfNorm)
    for i in range(len(dfTestNorm)):
        resultNo = ((1/(nolVar['x1'] * math.sqrt(2 * math.pi))) * math.exp(-(dfTestNorm['x1'][i] - nolMean['x1'])/(2 * nolVar['x1']**2)) *
                    (1/(nolVar['x2'] * math.sqrt(2 * math.pi))) * math.exp(-(dfTestNorm['x2'][i] - nolMean['x2'])/(2 * nolVar['x2']**2)) *
                    (1/(nolVar['x3'] * math.sqrt(2 * math.pi))) * math.exp(-(dfTestNorm['x3'][i] - nolMean['x3'])/(2 * nolVar['x3']**2)) *
                    (len(dfNol)/len(dfNorm)))
        resultYes = ((1/(satuVar['x1'] * math.sqrt(2 * math.pi))) * math.exp(-(dfTestNorm['x1'][i] - satuMean['x1'])/(2 * satuVar['x1']**2)) *
                    (1/(satuVar['x2'] * math.sqrt(2 * math.pi))) * math.exp(-(dfTestNorm['x2'][i] - satuMean['x2'])/(2 * satuVar['x2']**2)) *
                    (1/(satuVar['x3'] * math.sqrt(2 * math.pi))) * math.exp(-(dfTestNorm['x3'][i] - satuMean['x3'])/(2 * satuVar['x3']**2)) *
                    (len(dfSatu)/len(dfNorm)))
        if resultNo > resultYes:
            hasilBayes.append([dfTestNorm.loc[i, 'id'], 0])
        else:
            hasilBayes.append([dfTestNorm.loc[i, 'id'], 1])
    result = pd.DataFrame(data = hasilBayes, columns = ['id', 'y'])
```

Validation

Bayes Validation

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Accuracy	72%	83%	78%	60%	74%

Naïve Bayes

id	y
297	1
298	1
299	1
300	1
301	1
302	1
303	1
304	1
305	1
306	1

Thank You!