



## Hands-On

Hands-On ini digunakan pada kegiatan Microcredential Associate Data Scientist 2021

## Tugas Mandiri Pertemuan 16

Pertemuan 16 (enambelas) pada Microcredential Associate Data Scientist 2021 menyampaikan materi mengenai Membangun model: Evaluasi. silakan Anda kerjakan Latihan 1 s/d 5. Output yang anda lihat merupakan panduan yang dapat Anda ikuti dalam penulisan code :)

### Soal 1: Pemahaman Tentang Model Evaluasi

Jawab pertanyaan di bawah ini dengan bahasa masing-masing?

1. Apa perbedaan antara data latih, data validasi, dan data test?
2. Bagaimana cara kita menilai performa suatu model?
3. Apa itu Confusion Matrix? Jelaskan secara lengkap!
4. Apa itu Classification Report dari sklearn?

**Jawab:**

1. Data latih adalah data yang digunakan untuk melakukan training model, data validasi adalah data yang digunakan untuk melakukan validasi/asesmen model sementara dalam mencegah overfitting atau underfitting, data test adalah data yang digunakan untuk simulasi penggunaan model pada dunia nyata dan menghitung score model yang asli.
2. Untuk menilai performa model dilakukan dengan menghitung metrik evaluasi terhadap model tersebut, beberapa diantaranya terbagi menjadi
  - Klasifikasi: accuracy, precision, recall, F1-score, ROC, AUC, dan lainnya.
  - Regresi: MSE, MAE, R-Square, dan lainnya.
  - Klastering: Silhouette Coefficient, Davies-Bouldin Index, Dunn Index, dan lainnya.
3. Confusion matrix merupakan sebuah tabel evaluasi yang digunakan dalam model klasifikasi dimana isi dari tabel tersebut berupa nilai True Positif, False Positif, False Negatif, True Negatif terhadap nilai prediksi dengan nilai actual
4. Classification report menggambarkan laporan hasil klasifikasi dengan memberikan beberapa nilai metrik evaluasi terhadap model hasil training yang berupa nilai precision, recall, f1 score, accuracy dan lainnya pada setiap label yang ada.

### Soal 2: Aplikasi Model Evaluasi

Kali ini kita akan menggunakan data untuk memprediksi kelangsungan hidup pasien yang telah mengalami operasi payudara. Dengan informasi yang dimiliki terkait pasien, kita akan membuat model untuk memprediksi apakah pasien akan bertahan hidup dalam waktu lebih dari 5 tahun atau tidak.

Lebih Lengkapnya kalian bisa membaca informasi tentang dataset di link berikut:

<https://raw.githubusercontent.com/brownlee/Datasets/master/haberman.names>

Buat model Klasifikasi (Model/Algoritma Bebas) untuk memprediksi status pasien dengan ketentuan sebagai berikut:

1. Bagi kedua data ini menjadi data training dan data test dengan `test_size=0.25`.
2. Pelajari tentang metrics `roc_auc_score` kemudian buatlah model dan evaluasi dengan menggunakan teknik cross-validation dengan scoring '`roc_auc`'. Baca [https://scikit-learn.org/stable/modules/generated/sklearn.model\\_selection.cross\\_val\\_score.html](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html) untuk menggunakan metric `roc_auc` saat cross-validation.
3. Berapa score rata2 dari model dengan teknik cross-validation tersebut?
4. Prediksi data test dengan model yang telah kalian buat!
5. Bagaimana hasil confusion matrix dari hasil prediksi tersebut?
6. Bagaimana classification report dari hasil prediksi tersebut?
7. Seberapa baik model anda dalam memprediksi seorang pasien mempunyai status positive?
8. Seberapa baik model anda dalam memprediksi seorang pasien mempunyai status negatif?

## Load Dataset

```
In [29]: # import Library pandas
import pandas as pd

# Load Dataset
url = 'https://raw.githubusercontent.com/jbrownlee/Datasets/master/haberman.csv'
list_cols = ['Age', "Patient's Years", "N_positive_ax", "survival_status"]
df = pd.read_csv(url, names=list_cols)
```

```
In [30]: # tampilkan 5 baris awal dataset dengan function head()
df.head()
```

```
Out[30]:
```

	Age	Patient's Years	N_positive_ax	survival_status
0	30	64	1	1
1	30	62	3	1
2	30	65	0	1
3	31	59	2	1
4	31	65	4	1

```
In [31]: # hitung jumlah masing" data pada kolom survival_status
df['survival_status'].value_counts()
```

```
Out[31]:
```

1	225
2	81

Name: survival\_status, dtype: int64

## Build Model

```
In [32]: #import library train test split dan cross val
from sklearn.model_selection import train_test_split, cross_val_score

#import library Logistic regression
from sklearn.linear_model import LogisticRegression

#import library roc auc score
from sklearn.metrics import roc_auc_score

#import library scale
from sklearn.preprocessing import scale

#import library numpy
import numpy as np
```

```
In [33]: ## pemisahan feature dan target (data target : 'survival_status')
X = df.drop('survival_status', axis = 1)
Xs = scale(X)
y = df['survival_status']
```

### NO 1

```
In [34]: ## pemisahan variabel test dan train dari data Xs dan y
# test size= 25%, random state = 42, dan stratify = y
X_train, X_test, y_train, y_test = train_test_split(Xs, y, test_size=0.25, random_state=42)
```

```
In [35]: ## pembuatan objek model
model_logReg = LogisticRegression(random_state = 42)

## Latih model
model_logReg.fit(X_train, y_train)

## prediksi.
y_predict = model_logReg.predict(X_test)
```

### NO 2

```
In [36]: ## menghitung cross_val_score dengan scoring = 'roc_auc'
## parameter cv = 10
score = cross_val_score(model_logReg, X, y, scoring = 'roc_auc', cv = 10)
print(score)
```

```
[0.44021739 0.80978261 0.67391304 0.69021739 0.70380435 0.79292929
 0.875      0.62784091 0.67613636 0.61363636]
```

### NO 3

```
In [37]: # cetak rata-rata nilai rata-rata auc score
score.mean()
```

```
Out[37]: 0.6903477711901624
```

## NO 4

```
In [38]: # Prediksi data test dengan model yang telah kalian buat  
auc_score = roc_auc_score(y_test, y_predict)  
auc_score
```

```
Out[38]: 0.56818181818182
```

## NO 5

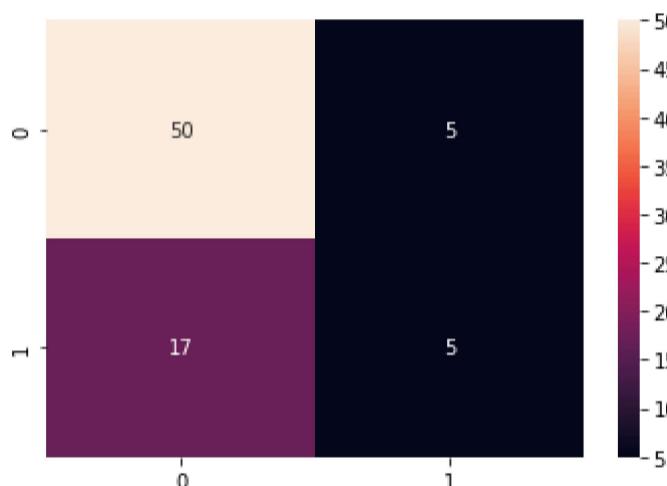
```
In [39]: # import Library confusion matrix dan classification report  
from sklearn.metrics import confusion_matrix, classification_report
```

```
In [40]: # apply confusion matrix dan cetak nilai confusion matrix  
cm = confusion_matrix(y_test, y_predict, labels = (1,2))  
cm
```

```
Out[40]: array([[50,  5],  
               [17,  5]], dtype=int64)
```

```
In [42]: # visualisasikan nilai confusion matrix ke dalam diagram heatmap  
import seaborn as sns  
  
sns.heatmap(cm, annot=True)
```

```
Out[42]: <AxesSubplot:>
```



## NO 6

```
In [43]: # cetak nilai classification_report  
print(classification_report(y_test, y_predict))
```

	precision	recall	f1-score	support
1	0.75	0.91	0.82	55
2	0.50	0.23	0.31	22
accuracy			0.71	77
macro avg	0.62	0.57	0.57	77
weighted avg	0.68	0.71	0.67	77

## NO 7

- Bagaimana hasil confusion matrix dari hasil prediksi tersebut?

Hasil confusion matrix cukup berat sebelah dimana nilai True Positif (label 1) sangat baik yaitu 50 buah sedangkan True Negatif (label 2) sangat tidak baik yaitu 5 buah, banyak nilai kesalahan klasifikasi terdapat pada hasil negatif / False Negatif yaitu 17 buah dan kesalahan pada hasil positif cukup sedikit hanya 5 buah

- Bagaimana classification report dari hasil prediksi tersebut?

Hasil classification report sama seperti confusion matrix dimana terdapat ketidakseimbangan antara nilai positif (label 1) dengan nilai negatif (label 2), karena untuk nilai positif terlihat sudah sangat baik dengan nilai f1-score 82% sedangkan nilai negatif terlihat buruk dengan nilai f1-score 31%. secara akurasi jumlah tersebut cukup pada rata-rata dengan nilai akurasi sebesar 71%

- Seberapa baik model anda dalam memprediksi seorang pasien mempunyai status positive? dari hasil classification\_report diatas

Berdasarkan classification report model hasil prediksi untuk pasien yang berstatus positif sudah cukup baik dimana nilai recall dapat dicapai sangat tinggi sebesar 91% yang artinya hanya sedikit nilai positif yang salah prediksi menjadi nilai negatif, dan nilai precision juga cukup baik yaitu 75% yang artinya kekeliruan hasil positif pada prediksi lebih banyak dari pada kesalahan prediksi hasil negatif

- Seberapa baik model anda dalam memprediksi seorang pasien mempunyai status negatif? dari hasil classification\_report diatas

Berdasarkan classification report model hasil prediksi untuk pasien yang berstatus negatif sangat buruk karena nilai recall yang diperoleh sangat rendah yaitu 23% yang artinya sangat banyak kesalahan prediksi yang harus nya negatif tapi ternilai positif, dan pada nilai precision juga buruk yaitu 50% yang artinya terdapat jumlah perbandingan yang sama antara aktualnya positif tapi prediksinya jadi negatif

## Soal 3: Pemahaman Tentang Model Selection

Jelaskan dengan bahasa sendiri!

1. Apa itu Bias dan Variance?
2. Apa itu Overfitting dan Underfitting?
3. Apa yang bisa kita lakukan untuk mengatur kompleksitas dari model?
4. Bagaimana model yang baik?
5. Kapan kita menggunakan GridSearchCV dan kapan menggunakan RandomizedSearchCV?

Jawab

1. Bias adalah banyaknya perbedaan antara rata-rata prediksi dengan prediksi yang benar, Bias tinggi menandakan bahwa model bisa underfitting, Bias rendah bisa menandakan bahwa model overfitting. Variance adalah banyaknya hasil data prediksi yang berganti jika training data set dirubah, Varians tinggi bisa menandakan bahwa model overfitting, Varians rendah bisa menandakan bahwa model kurang pas.
2. Overfitting adalah suatu keadaan dimana data yang digunakan untuk pelatihan itu adalah yang "terbaik", biasanya hal ini menjadi akurasi data training yang sangat tinggi namun akurasi data testing sangat kecil atau jauh berbeda dengan hasil akurasi data training. Underfitting adalah keadaan dimana model pelatihan data yang dibuat tidak mewakilkan keseluruhan data yang akan digunakan nantinya. Sehingga menghasilkan performa yang buruk dalam pelatihan data.
3. Beberapa hal yang dapat dilakukan yaitu dengan menggunakan data yang bagus (telah melakukan proses data preparasi yang baik), memilih fitur yang bagus, memilih model yang sesuai dengan kasus, melakukan proses training yang cukup untuk mewakili hasil aktual, dan menggunakan parameter-parameter yang pas.
4. Model yang baik adalah model yang dapat menyeimbangkan hasil performa yang tinggi tanpa adanya underfitting maupun overfitting, sehingga hasil performa training dapat mencapai nilai yang tinggi dan performa testing memiliki nilai yang tinggi.
5. GridSearchCV digunakan sangat baik ketika parameter yang dicoba tidak terlalu banyak dan memiliki waktu training yang cepat. Sedangkan RandomizedSearchCV sebaliknya yaitu digunakan sangat baik ketika parameter yang dicoba sangat banyak serta waktu training yang lambat.

## Soal 4: Aplikasi Model Selection

1. Bagi kedua data berikut ini menjadi data training dan data test dengan test\_size=0.25.
2. Import library KNN dan GridSearchCV.
3. Gunakan algoritma KNN dan fungsi GridSearchCV untuk hyperparameter tuning dan model selection.
4. jumlah fold bebas!, gunakan scoring 'roc\_auc'
5. Definisikan kombinasi hyperparameter untuk model selection dengan GridSearchCV. kombinasi Hyperparameter bebas, baca lagi dokumentasi KNN di link berikut <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html> untuk memahami lagi jenis2 hyperparameter di algorithma KNN.
6. Latih model terhadap data training.
7. Apa hyperparameter terbaik untuk kombinasi hyperparameter kalian?
8. Berapa score validasi terbaik dari model tersebut?
9. Prediksi probabilitasi output dari model yang telah dibuat terhadap data test. note : gunakan method .predict\_proba() untuk menghasilkan output probabilitas
10. Berapa nilai score roc\_auc untuk data test? (y\_predict)
11. Apakah model anda termasuk baik, overfitting, atau underfitting?

### Load Dataset

In [52]:

```
# import Library pandas
import pandas as pd

# Load Dataset
url = 'https://raw.githubusercontent.com/jbrownlee/Datasets/master/haberman.csv'
list_cols = ['Age', "Patient's Years", "N_positive_ax", "survival_status"]
df2 = pd.read_csv(url, names=list_cols)
```

In [53]:

```
# tampilkan 5 baris awal dataset dengan function head()
df2.head()
```

Out[53]:

	Age	Patient's Years	N_positive_ax	survival_status
0	30	64	1	1
1	30	62	3	1
2	30	65	0	1
3	31	59	2	1
4	31	65	4	1

In [54]:

```
# hitung jumlah masing" data pada kolom survival_status
df2['survival_status'].value_counts()
```

```
Out[54]: 1    225
2     81
Name: survival_status, dtype: int64
```

## NO 1

```
In [55]: # 1. pembagian variabel train dan test
# test size= 25%, random state = 42, dan stratify = y
X = df2.drop('survival_status', axis = 1)
y = df2['survival_status']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```

## NO 2

```
In [56]: # 2. import library KNN dan GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
```

## NO 3 - 6

```
In [57]: # 3. tuning hyperparameter dengan GridSearchCV (parameter cv=10)
## build model KNN
model_knn = KNeighborsClassifier()
param_grid = {'n_neighbors' : np.arange(3,51), 'weights' : ['uniform','distance']}
gscv = GridSearchCV(model_knn, param_grid, scoring='roc_auc', cv = 10)
gscv.fit(X_train, y_train)
```

```
Out[57]: GridSearchCV(cv=10, estimator=KNeighborsClassifier(),
param_grid={'n_neighbors': array([ 3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36,
37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50]),
'weights': ['uniform', 'distance']},
scoring='roc_auc')
```

## NO 7

```
In [58]: # 7. parameter terbaik
gscv.best_params_

Out[58]: {'n_neighbors': 12, 'weights': 'uniform'}
```

## NO 8

```
In [59]: # 8. score validasi terbaik
gscv.best_score_
```

```
Out[59]: 0.7316666666666667
```

## NO 9

```
In [62]: # 9. prediksi probabilitas masing-masing data test
y_predict = gscv.predict_proba(X_test)
y_predict
```

```
Out[62]: array([[0.83333333, 0.16666667],
[1.        , 0.        ],
[0.91666667, 0.08333333],
[0.91666667, 0.08333333],
[0.66666667, 0.33333333],
[0.58333333, 0.41666667],
[0.83333333, 0.16666667],
[1.        , 0.        ],
[0.33333333, 0.66666667],
[0.75        , 0.25        ],
[0.91666667, 0.08333333],
[0.83333333, 0.16666667],
[0.5        , 0.5        ],
[0.25        , 0.75        ],
[0.83333333, 0.16666667],
[0.91666667, 0.08333333],
[1.        , 0.        ],
[0.83333333, 0.16666667],
[0.58333333, 0.41666667],
[0.75        , 0.25        ],
[0.75        , 0.25        ],
[1.        , 0.        ],
[0.91666667, 0.08333333],
[1.        , 0.        ],
[0.75        , 0.25        ],
[1.        , 0.        ],
[0.83333333, 0.16666667],
[0.58333333, 0.41666667],
[0.58333333, 0.41666667],
[1.        , 0.        ],
[0.41666667, 0.58333333],
```

```
[1.        , 0.        ],
[0.66666667, 0.33333333],
[0.75      , 0.25      ],
[0.5       , 0.5       ],
[0.66666667, 0.33333333],
[0.66666667, 0.33333333],
[0.75      , 0.25      ],
[0.83333333, 0.16666667],
[0.5       , 0.5       ],
[0.66666667, 0.33333333],
[0.33333333, 0.66666667],
[0.91666667, 0.08333333],
[1.        , 0.        ],
[0.83333333, 0.16666667],
[0.83333333, 0.16666667],
[0.66666667, 0.33333333],
[0.83333333, 0.16666667],
[0.75      , 0.25      ],
[0.25      , 0.75      ],
[0.83333333, 0.16666667],
[0.83333333, 0.16666667],
[0.58333333, 0.41666667],
[1.        , 0.        ],
[0.33333333, 0.66666667],
[0.58333333, 0.41666667],
[0.75      , 0.25      ],
[0.75      , 0.25      ],
[1.        , 0.        ],
[0.91666667, 0.08333333],
[0.58333333, 0.41666667],
[0.91666667, 0.08333333],
[0.75      , 0.25      ],
[0.33333333, 0.66666667],
[1.        , 0.        ],
[0.5       , 0.5       ],
[0.75      , 0.25      ],
[1.        , 0.        ],
[0.5       , 0.5       ],
[0.83333333, 0.16666667],
[0.75      , 0.25      ],
[0.33333333, 0.66666667],
[0.91666667, 0.08333333],
[0.83333333, 0.16666667],
[0.83333333, 0.16666667],
[1.        , 0.        ]])
```

In [63]: `# nilai rata-rata probabilitas data test  
y_predict.mean()`

Out[63]: 0.5

## NO 10

In [64]: `# 10. nilai score roc_auc  
kurang_5th = y_predict[:,1]  
print(kurang_5th)`

```
[0.16666667 0.        0.08333333 0.08333333 0.33333333 0.41666667
 0.16666667 0.        0.66666667 0.25      0.08333333 0.16666667
 0.5       0.75      0.16666667 0.08333333 0.        0.16666667
 0.41666667 0.25      0.25      0.        0.08333333 0.
 0.25      0.        0.        0.16666667 0.41666667 0.41666667
 0.        0.58333333 0.        0.33333333 0.25      0.5
 0.33333333 0.33333333 0.25      0.16666667 0.5      0.33333333
 0.66666667 0.08333333 0.        0.16666667 0.16666667 0.33333333
 0.16666667 0.25      0.75      0.16666667 0.16666667 0.41666667
 0.        0.66666667 0.41666667 0.25      0.25      0.
 0.08333333 0.41666667 0.08333333 0.25      0.66666667 0.
 0.5       0.25      0.        0.5      0.16666667 0.25
 0.66666667 0.08333333 0.16666667 0.16666667 0.        ]
```

## NO 11

**Jawab**

In [70]: `roc_auc_score(y_test, kurang_5th)`

Out[70]: 0.6553719008264463

Hasil performa metrik evaluasi pada data training sebesar 73.2% dan data testing sebesar 65.5%. Sehingga kesimpulannya model yang dibangun cukup underfitting dimana performa yang diperoleh cukup kecil baik dari data training maupun data testing

## Soal 5:

- Ulangi tahap di atas (soal 4, no 1 - 8) namun kali ini menggunakan algoritma DecisionTreeClassifier dan kalian bisa menggunakan RandomizedSearchCV apabila process training lama. pelajari algoritma DecisionTreeClassifier di linkberikut: <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html?highlight=decisiontreeclassifier#sklearn.tree.DecisionTreeClassifier>

2. Bandingkan scorenya dengan Algoritma KNN, mana yang lebih baik?

Note : Data Science adalah experiment, sangat di dimungkinkan memerlukan beberapa kali percobaan untuk mendapatkan hasil yang terbaik! Happy Coding :)

## NO 1

```
In [71]: # 1. import algoritma DecisionTreeClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import RandomizedSearchCV

In [90]: # Build model decision tree classifier
model_tree = DecisionTreeClassifier()
params = {'criterion': ['entropy', 'gini'], 'splitter': ['best', 'random'],
          'min_samples_split': np.arange(2,50)}
gscv = GridSearchCV(model_tree, param_grid = params, cv = 10, scoring = 'roc_auc')
gscv.fit(X_train, y_train)

Out[90]: GridSearchCV(cv=10, estimator=DecisionTreeClassifier(),
                     param_grid={'criterion': ['entropy', 'gini'],
                                 'min_samples_split': array([ 2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
                                19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
                                36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49]),
                                 'splitter': ['best', 'random']},
                     scoring='roc_auc')

In [91]: # parameter terbaik
gscv.best_params_

Out[91]: {'criterion': 'gini', 'min_samples_split': 20, 'splitter': 'random'}

In [92]: # score validasi terbaik
gscv.best_score_

Out[92]: 0.7433333333333334
```

## NO 2

Jawab

```
In [93]: y_predict = gscv.predict_proba(X_test)
y_predict
```

```
Out[93]: array([[0.75      , 0.25      ],
   [0.94117647, 0.05882353],
   [0.71428571, 0.28571429],
   [0.94117647, 0.05882353],
   [0.71428571, 0.28571429],
   [1.        , 0.        ],
   [0.81818182, 0.18181818],
   [0.94117647, 0.05882353],
   [0.25      , 0.75      ],
   [0.88888889, 0.11111111],
   [1.        , 0.        ],
   [0.8        , 0.2        ],
   [0.8        , 0.2        ],
   [0.4        , 0.6        ],
   [1.        , 0.        ],
   [0.75      , 0.25      ],
   [0.71428571, 0.28571429],
   [1.        , 0.        ],
   [0.25      , 0.75      ],
   [0.81818182, 0.18181818],
   [1.        , 0.        ],
   [0.85714286, 0.14285714],
   [0.94117647, 0.05882353],
   [1.        , 0.        ],
   [0.4375    , 0.5625    ],
   [0.94117647, 0.05882353],
   [1.        , 0.        ],
   [0.52941176, 0.47058824],
   [0.66666667, 0.33333333],
   [0.52941176, 0.47058824],
   [0.71428571, 0.28571429],
   [0.33333333, 0.66666667],
   [1.        , 0.        ],
   [0.88888889, 0.11111111],
   [0.88888889, 0.11111111],
   [0.66666667, 0.33333333],
   [1.        , 0.        ],
   [1.        , 0.        ],
   [0.88888889, 0.11111111],
   [0.71428571, 0.28571429],
   [0.4        , 0.6        ],
   [0.71428571, 0.28571429],
   [0.4        , 0.6        ],
   [0.81818182, 0.18181818],
```

```
[1.      , 0.      ],
[1.      , 0.      ],
[1.      , 0.      ],
[1.      , 0.      ],
[0.66666667, 0.33333333],
[1.      , 0.      ],
[0.4     , 0.6    ],
[0.66666667, 0.33333333],
[0.81818182, 0.18181818],
[0.8     , 0.2    ],
[1.      , 0.      ],
[0.33333333, 0.66666667],
[0.52941176, 0.47058824],
[1.      , 0.      ],
[0.4375  , 0.5625 ],
[1.      , 0.      ],
[0.94117647, 0.05882353],
[0.66666667, 0.33333333],
[0.4375  , 0.5625 ],
[0.75    , 0.25   ],
[0.4     , 0.6    ],
[0.94117647, 0.05882353],
[0.4     , 0.6    ],
[0.71428571, 0.28571429],
[1.      , 0.      ],
[0.52941176, 0.47058824],
[0.88888889, 0.11111111],
[1.      , 0.      ],
[0.4     , 0.6    ],
[0.71428571, 0.28571429],
[0.94117647, 0.05882353],
[0.71428571, 0.28571429],
[1.      , 0.      ]])
```

```
In [94]: y_predict.mean()
```

```
Out[94]: 0.5
```

```
In [95]: kurang_5th = y_predict[:,1]
print(kurang_5th)
```

```
[0.25    0.05882353 0.28571429 0.05882353 0.28571429 0.
 0.18181818 0.05882353 0.75      0.11111111 0.        0.2
 0.2     0.6      0.        0.25      0.28571429 0.
 0.75    0.18181818 0.        0.14285714 0.05882353 0.
 0.5625  0.05882353 0.        0.47058824 0.33333333 0.47058824
 0.28571429 0.66666667 0.        0.11111111 0.11111111 0.33333333
 0.        0.        0.11111111 0.28571429 0.6      0.28571429
 0.6     0.18181818 0.        0.        0.        0.
 0.33333333 0.        0.6      0.33333333 0.18181818 0.2
 0.        0.66666667 0.47058824 0.        0.5625   0.
 0.05882353 0.33333333 0.5625   0.25      0.6      0.05882353
 0.6     0.28571429 0.        0.47058824 0.11111111 0.
 0.6     0.28571429 0.05882353 0.28571429 0.        ]
```

```
In [96]: roc_auc_score(y_test, kurang_5th)
```

```
Out[96]: 0.6669421487603306
```

Kesimpulan :

- Performa KNN pada Data Training : 73.2%
- Performa KNN pada Data Testing : 65.5%
- Performa Decision Tree Pada Data Training : 74%
- Performa Decision Tree Pada Data Testing : 66.7%

Berdasarkan hasil yang diperoleh terlihat bahwa model yang dibangun oleh Decision Tree sedikit lebih baik dibandingkan KNN meskipun hanya berbeda 1%