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.Perbedaan Data Training, Validasi dan Test:

A. Data latih atau Training digunakan untuk training model

B. Data Validasi digunakan untuk digunakan untuk proses validasi model dan mencegah overfitting.

Dataset di input ke sistem untuk menghitung loss function, tanpa melakukan update bias dan weight. Setiap epoch atau iterasi, proses training dan validasi dilakukan beruntutan. Setiap selesai melakukan training, dilanjutkan dengan proses validasi. JIka nilai loss function dari proses validasi naik, training model dihentikan.

C. Data Test merupakan data tester yang digunakan untuk testing model, sebagai simulasi penggunaan model pada dunia nyata. Data testing tidak boleh pernah dilihat oleh model sebelumnya.

2.Performa model -ketika data telah melalui ketiga tahap diatas, dan mendapatkan akurasi diatas 80% maka dapat disimpulkan data sudah baik, semakin tinggi akurasi maka semakin baik model yang dihasilkan.

3.Pengertian Confusion Matrix

Confusion matrix adalah sebuah tabel yang sering digunakan untuk mengukur kinerja dari model klasifikasi di machine learning. Tabel ini menggambarkan lebih detail tentang jumlah data yang diklasifikasikan dengan benar maupun salah.

Confusion matrix adalah salah satu tools analitik prediktif yang menampilkan dan membandingkan nilai aktual atau nilai sebenarnya dengan nilai hasil prediksi model yang dapat digunakan untuk menghasilkan metrik evaluasi seperti Accuracy (akurasi), Precision, Recall, dan F1-Score atau F-Measure.

Ada empat nilai yang dihasilkan di dalam tabel confusion matrix, di antaranya True Positive (TP), False Positive (FP), False Negative (FN), dan True Negative (TN).

True Positive (TP) : Jumlah data yang bernilai Positif dan diprediksi benar sebagai Positif.

False Positive (FP) : Jumlah data yang bernilai Negatif tetapi diprediksi sebagai Positif.

False Negative (FN) : Jumlah data yang bernilai Positif tetapi diprediksi sebagai Negatif.

True Negative (TN) : Jumlah data yang bernilai Negatif dan diprediksi benar sebagai Negatif.

4.Pengertian Classification Report dari sklearn

Classification Report menampilkan presisi, ingatan, F1, dan skor dukungan untuk model. Untuk mendukung interpretasi dan deteksi masalah yang lebih mudah, laporan ini mengintegrasikan skor numerik dengan peta panas berkode warna. Semua peta panas berada dalam kisaran (0.0, 1.0) untuk memudahkan perbandingan model klasifikasi di berbagai laporan klasifikasi.

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/jbrownlee/Datasets/master/haberman.names
(https://raw.githubusercontent.com/jbrownlee/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. Pelajar 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 [27]:
```
# 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 [38]:
```
# tampilkan 5 baris awal dataset dengan function head()
df.head()
```

Out[38]:

	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 [39]:
```
# hitung jumlah masing" data pada kolom survival_status
df['survival_status'].value_counts()
```

Out[39]:
```
1    225
2     81
Name: survival_status, dtype: int64
```

Build Model

In [64]:
```
#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

import seaborn as sns
import matplotlib.pyplot as plt
```

In [65]:
```
## 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 [66]:
```
## pemisahan variabel test dan train dari data Xs dan y
# test size= 25%, random state = 42, dan stratify = y

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, randc
```

In [67]:
```
## 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 [68]:
```
## 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 [69]:
```
# cetak rata-rata nilai rata-rata auc score
score.mean()
```

Out[69]:
```
0.6903477711901624
```

NO 4

In [70]:
```
# Prediksi data test dengan model yang telah kalian buat
auc_score = roc_auc_score(y_test, y_predict)
auc_score
```

Out[70]:
```
0.5681818181818182
```

NO 5

In [71]:
```
# import library confusion matrix dan classification report
from sklearn.metrics import confusion_matrix

from sklearn.metrics import classification_report
```

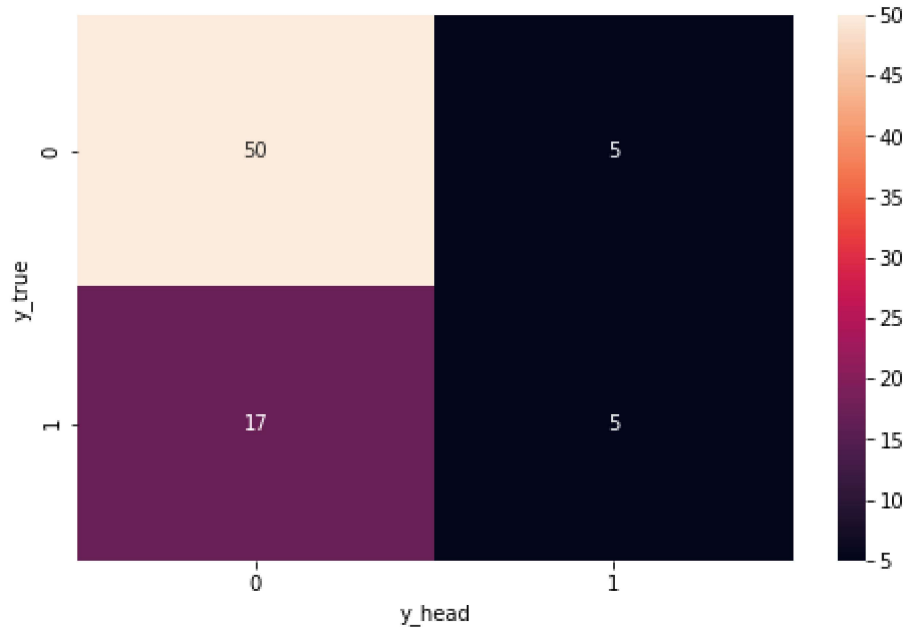
In [72]:
```
conda update -c conda-forge scikit-learn
```

```
Collecting package metadata (current_repodata.json): ...working... done
Solving environment: ...working...

Updating scikit-learn is constricted by

anaconda -> requires scikit-learn==0.24.1=py38hf11a4ad_0

If you are sure you want an update of your package either try `conda update --a
ll` or install a specific version of the package you want using `conda install
<pkg>=<version>`

done

# All requested packages already installed.


Note: you may need to restart the kernel to use updated packages.
```

In [73]:
```
# apply confusion matrix dan cetak nilai confusion matrix
cm = confusion_matrix(y_test,y_predict, labels = (1,2))
cm
```

Out[73]:
```
array([[50,  5],
       [17,  5]], dtype=int64)
```

In [75]:
```
# visualisasikan nilai confusion matrix ke dalam diagram heatmap
f, ax = plt.subplots(figsize=(8,5))
sns.heatmap(confusion_matrix(y_test, y_predict), annot=True, fmt=".0f", ax=ax)
plt.xlabel("y_head")
plt.ylabel("y_true")
plt.show()
```



NO 6

In [77]:
```
# cetak nilai classification_report
from sklearn.metrics import 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 & 8

- Bagaimana hasil confusion matrix dari hasil prediksi tersebut?
  Berdasarkan hasil Confusion matrix diatas menunjukkan bahwa hasil y_test lebih besar dibandingkan dengan y_predict

- Bagaimana classification report dari hasil prediksi tersebut?
  Berdasarkan hasil classification report, data diatas menunjukkan bahwa percobaan pertama lebih tinggi dibanding percobaan kedua. Dengan accuracy f1-score bernilai 0,71.Dengan rata-rata besaran untuk precision diangka 0,62. Recall & f1-score : 0,57 dan rata-rata keseimbangan diangka 0,68 untuk precision dan 0,71 untuk recall serta f1-score diangka 0,67.

- Seberapa baik model anda dalam memprediksi seorang pasien mempunyai status positive? dari hasil classification_report diatas
  Dari hasil akurasi yang cukup rendah, diperlukan penggunaan metode lain untuk meningkatkan nilai akurasi seorang pasien. Dengan nilai status positive 17

- Seberapa baik model anda dalam memprediksi seorang pasien mempunyai status negatif? dari hasil classification_report diatas
  Dari hasil akurasi yang cukup rendah, diperlukan penggunaan metode lain untuk meningkatkan nilai akurasi seorang pasien.Dengan nilai status negative 50

Soal 3: Pemahaman Tentang Model Selection

Jelaskan dengan bahasa sendiri!

1. Apa itu Bias dan Variance?

Bias adalah perbedaan antara rata-rata hasil prediksi dari model ML yang kita develop dengan data nilai yang sebenarnya. Bias yang tinggi dikarenakan dalam pembangunan model ML, dilakukan terlalu sederhan (oversimplified). Faktor penyebab lain dikarenakan model ML yang di develop terlalu tidak terlalu berinteraksi dengan training data.Bias seringkali terjadi dalam development sistem machine learning.

Variance adalah variabel dari prediksi model untuk data tertentu dimana memberikan kita informasi perserbaran data kita. Model yang memiliki variance tinggi sangat memperhatikan hanya pada train data. High variance model, perform baik di train data. Tetapi jika disuguhkan data baru yang belum pernah ditemukan di train data. Model tersebut tidak dapat mengeneralisasikan secara baik dari identifikasi data baru tersebut. Alhasil model memprediksi dengan keliru.

2. Apa itu Overfitting dan Underfitting?

Overfitting adalah suatu keadaan dimana data yang digunakan untuk pelatihan itu adalah yang "terbaik". Sehingga apabila dilakukan tes dengan menggunakan data yang berbeda dapat mengurangi akurasi (hasil yang dibuat tidak sesuai yang diharapkan). Overfitting dapat terjadi ketika beberapa batasan didasarkan pada sifat khusus yang tidak membuat perbedaan pada data. Selain itu duplikasi data minor yang berlebihan juga dapat mengakibatkan terjadinya overfitting.

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. Underfitting terjadi karena model masih mempelajari struktur dari data. Hasilnya, tree bekerja dengan buruk pada masa pelatihan dan tes. Sebagaimana banyaknya node

dalam pohon keputusan meningkat, tree memiliki galat pelatihan dan tes yang lebih kecil. Pada saat tree berukuran sangat besar, tingkat terjadinya galat tes mulai meningkat walaupun tingkat galat pelatihannya terus menurun.

3. Apa yang bisa kita lakukan untuk mengatur kompleksitas dari model?

Penjabaran lebih komleks pada klasifikasi data untuk menghasil predict yang lebih teratur.

4. Bagaimana model yang baik?

Model yang memiliki tingkat akurasi yang tinggi disertai dengan nilai yang tidak Overfitting ataupun Underfitting. Sehingga dapat dikatakan hasil test dan prediksi seimbang.

5. Kapan kita menggunakan GridSearchcv dan kapan menggunakan RandomizedSearchCV?

Grid Search adalah metode yang efektif untuk menyesuaikan parameter dalam supervised learning dan meningkatkan performa generalisasi model.

RandomizedSearchCV sangat berguna ketika data unsupervised learning serta memiliki banyak parameter untuk dicoba dan waktu pelatihannya sangat lama.

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 (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 di buat 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, overtting, atau underfitting?

Load Dataset

In [78]:
```
# 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 [79]:
```
# tampilkan 5 baris awal dataset dengan function head()
df2.head()
```

Out[79]:

	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 [80]:
```
# hitung jumlah masing" data pada kolom survival_status
df2['survival_status'].value_counts()
```

Out[80]:
```
1    225
2     81
Name: survival_status, dtype: int64
```

NO 1

In [81]:
```
# 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']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, rando
```

NO 2

In [82]:
```
# 2. import library KNN dan GridSearchCv
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
```

NO 3 - 6

In [83]:
```
# 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[83]:
```
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 [84]:
```
# 7. parameter terbaik
gscv.best_params_
```

Out[84]:
```
{'n_neighbors': 12, 'weights': 'uniform'}
```

NO 8

In [85]:
```
# 8. score validasi terbaik
gscv.best_score_
```

Out[85]:
```
0.7316666666666667
```

NO 9

In [86]:
```
# 9. prediksikan probabilitas masing-masing data test
y_predict = gscv.predict_proba(X_test)
y_predict
```

Out[86]:
```
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      ],
```

In [90]:
```
# nilai rata-rata probabilitas data test
y_predict.mean()
```

Out[90]:
```
0.5
```

In [119]:
```
y_train
```

Out[119]:
```
240    2
204    1
275    1
165    1
220    1
      ..
188    1
71     1
106    1
270    1
102    1
Name: survival_status, Length: 229, dtype: int64
```

In [118]:
```
y_test
```

Out[118]:
```
182    1
154    1
111    1
203    1
60     1
      ..
260    2
56     1
247    1
114    2
132    1
Name: survival_status, Length: 77, dtype: int64
```

In [125]:
```
x = np.linspace(0, 2*np.pi, 8)
y = np.sin(x) + np.random.normal(0, 0.4, 8)
```

NO 10

10. nilai score roc_auc

kurang_5th = roc_auc_score[:,1] print(kurang_5th)

11. Pada hasil pengerjaan, data termasuk Overfitting karena tidak menunjukan hasil secara akurat

Soal 5:

1. 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 (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 [105]:
```
# 1. import algoritma DecisionTreeClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import RandomizedSearchCV
```

In [106]:
```
# 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[106]:
```
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 [107]:
```
# parameter terbaik
gscv.best_params_
```

Out[107]:
```
{'criterion': 'gini', 'min_samples_split': 22, 'splitter': 'random'}
```

In [108]:
```
# score validasi terbaik
gscv.predict_proba(X_test)
```

Out[108]:
```
array([[0.9       , 0.1       ],
       [1.        , 0.        ],
       [1.        , 0.        ],
       [0.9       , 0.1       ],
       [0.89473684, 0.10526316],
       [0.        , 1.        ],
       [1.        , 0.        ],
       [1.        , 0.        ],
       [0.44444444, 0.55555556],
       [1.        , 0.        ],
       [1.        , 0.        ],
       [0.66666667, 0.33333333],
       [1.        , 0.        ],
       [0.25      , 0.75      ],
       [0.66666667, 0.33333333],
       [0.89473684, 0.10526316],
       [1.        , 0.        ],
       [1.        , 0.        ],
       [0.44444444, 0.55555556],
       [1.        , 0.        ],
       [0.9       , 0.1       ],
       [0.89473684, 0.10526316],
       [0.66666667, 0.33333333],
       [1.        , 0.        ],
       [0.6       , 0.4       ],
       [0.9       , 0.1       ],
       [1.        , 0.        ],
       [0.6       , 0.4       ],
       [0.71428571, 0.28571429],
       [1.        , 0.        ],
       [1.        , 0.        ],
       [0.89473684, 0.10526316],
       [1.        , 0.        ],
       [0.89473684, 0.10526316],
       [0.66666667, 0.33333333],
       [0.71428571, 0.28571429],
       [0.61538462, 0.38461538],
       [0.66666667, 0.33333333],
       [0.66666667, 0.33333333],
       [1.        , 0.        ],
       [0.2       , 0.8       ],
       [0.61538462, 0.38461538],
       [0.2       , 0.8       ],
       [1.        , 0.        ],
       [0.89473684, 0.10526316],
       [0.66666667, 0.33333333],
       [0.89473684, 0.10526316],
       [0.61538462, 0.38461538],
       [0.5       , 0.5       ],
       [0.9       , 0.1       ],
       [0.44444444, 0.55555556],
       [0.71428571, 0.28571429],
       [0.61538462, 0.38461538],
```

```
[1.        , 0.        ],
[0.9       , 0.1       ],
[0.44444444, 0.55555556],
[0.44444444, 0.55555556],
[0.6       , 0.4       ],
[0.5       , 0.5       ],
[0.9       , 0.1       ],
[0.9       , 0.1       ],
[0.71428571, 0.28571429],
[0.6       , 0.4       ],
[0.9       , 0.1       ],
[0.25      , 0.75      ],
[1.        , 0.        ],
[0.25      , 0.75      ],
[0.61538462, 0.38461538],
[0.89473684, 0.10526316],
[0.6       , 0.4       ],
[0.66666667, 0.33333333],
[1.        , 0.        ],
[0.25      , 0.75      ],
[1.        , 0.        ],
[0.66666667, 0.33333333],
[1.        , 0.        ],
[1.        , 0.        ]])
```