Tugas Kecil 1 IF3270 Eksplorasi Library Algoritma Pembelajaran pada Jupyter Notebook

Anggota:

- 13520138 Gerald Abraham Sianturi
- 13520162 Daffa Romyz Aufa

Setup Pustaka

```
import pandas as pd
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
```

Setup dataframe

Dataframe keseluruhan

```
cancer_data = load_breast_cancer(return_X_y=True, as_frame=True)
X, y = cancer_data
df_join = pd.concat([X, y], axis=1)
df_join
```

```
     mean radius  mean texture  mean perimeter  mean area  mean
smoothness  \
0          17.99         10.38          122.80     1001.0
0.11840
1          20.57         17.77          132.90     1326.0
0.08474
2          19.69         21.25          130.00     1203.0
0.10960
3          11.42         20.38           77.58      386.1
0.14250
4          20.29         14.34          135.10     1297.0
0.10030
..           ...           ...             ...        ...
...
564        21.56         22.39          142.00     1479.0
0.11100
565        20.13         28.25          131.20     1261.0
0.09780
566        16.60         28.08          108.30      858.1
0.08455
567        20.60         29.33          140.10     1265.0
0.11780
568         7.76         24.54           47.92      181.0
0.05263

     mean compactness  mean concavity  mean concave points  mean
```

```
symmetry  \
0             0.27760         0.30010              0.14710
0.2419
1             0.07864         0.08690              0.07017
0.1812
2             0.15990         0.19740              0.12790
0.2069
3             0.28390         0.24140              0.10520
0.2597
4             0.13280         0.19800              0.10430
0.1809
..                ...             ...                  ...
...
564           0.11590         0.24390              0.13890
0.1726
565           0.10340         0.14400              0.09791
0.1752
566           0.10230         0.09251              0.05302
0.1590
567           0.27700         0.35140              0.15200
0.2397
568           0.04362         0.00000              0.00000
0.1587

     mean fractal dimension  ...  worst texture  worst perimeter
worst area  \
0                    0.07871  ...          17.33           184.60
2019.0
1                    0.05667  ...          23.41           158.80
1956.0
2                    0.05999  ...          25.53           152.50
1709.0
3                    0.09744  ...          26.50            98.87
567.7
4                    0.05883  ...          16.67           152.20
1575.0
..                       ...  ...            ...              ...
...
564                  0.05623  ...          26.40           166.10
2027.0
565                  0.05533  ...          38.25           155.00
1731.0
566                  0.05648  ...          34.12           126.70
1124.0
567                  0.07016  ...          39.42           184.60
1821.0
568                  0.05884  ...          30.37            59.16
268.6

     worst smoothness  worst compactness  worst concavity  \
```

```
0             0.16220           0.66560           0.7119
1             0.12380           0.18660           0.2416
2             0.14440           0.42450           0.4504
3             0.20980           0.86630           0.6869
4             0.13740           0.20500           0.4000
..                ...               ...              ...
564           0.14100           0.21130           0.4107
565           0.11660           0.19220           0.3215
566           0.11390           0.30940           0.3403
567           0.16500           0.86810           0.9387
568           0.08996           0.06444           0.0000

     worst concave points  worst symmetry  worst fractal dimension
target
0                  0.2654          0.4601                  0.11890
0
1                  0.1860          0.2750                  0.08902
0
2                  0.2430          0.3613                  0.08758
0
3                  0.2575          0.6638                  0.17300
0
4                  0.1625          0.2364                  0.07678
0
..                    ...             ...                      ...
...
564                0.2216          0.2060                  0.07115
0
565                0.1628          0.2572                  0.06637
0
566                0.1418          0.2218                  0.07820
0
567                0.2650          0.4087                  0.12400
0
568                0.0000          0.2871                  0.07039
1

[569 rows x 31 columns]
```

Exploratory Data Analysis

Features list

```
features = list(cancer_data[0])
print(f"{features}\n")
print(f"Banyak fitur: {len(features)}")
```

```
['mean radius', 'mean texture', 'mean perimeter', 'mean area', 'mean
smoothness', 'mean compactness', 'mean concavity', 'mean concave
points', 'mean symmetry', 'mean fractal dimension', 'radius error',
'texture error', 'perimeter error', 'area error', 'smoothness error',
```

```
'compactness error', 'concavity error', 'concave points error',
'symmetry error', 'fractal dimension error', 'worst radius', 'worst
texture', 'worst perimeter', 'worst area', 'worst smoothness', 'worst
compactness', 'worst concavity', 'worst concave points', 'worst
symmetry', 'worst fractal dimension']

Banyak fitur: 30
```

Feature data type

```
df_join.dtypes
```

```
mean radius                float64
mean texture               float64
mean perimeter             float64
mean area                  float64
mean smoothness            float64
mean compactness           float64
mean concavity             float64
mean concave points        float64
mean symmetry              float64
mean fractal dimension     float64
radius error               float64
texture error              float64
perimeter error            float64
area error                 float64
smoothness error           float64
compactness error          float64
concavity error            float64
concave points error       float64
symmetry error             float64
fractal dimension error    float64
worst radius               float64
worst texture              float64
worst perimeter            float64
worst area                 float64
worst smoothness           float64
worst compactness          float64
worst concavity            float64
worst concave points       float64
worst symmetry             float64
worst fractal dimension    float64
target                       int32
dtype: object
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
train_size=0.8, test_size=0.2, random_state=10)
df_train = pd.concat([X_train, y_train], axis=1)
df_test = pd.concat([X_test, y_test], axis=1)
```

Statistical summary

```
df_join.describe()
```

```
       mean radius  mean texture  mean perimeter    mean area  \
count   569.000000    569.000000      569.000000   569.000000
mean     14.127292     19.289649       91.969033   654.889104
std       3.524049      4.301036       24.298981   351.914129
min       6.981000      9.710000       43.790000   143.500000
25%      11.700000     16.170000       75.170000   420.300000
50%      13.370000     18.840000       86.240000   551.100000
75%      15.780000     21.800000      104.100000   782.700000
max      28.110000     39.280000      188.500000  2501.000000

       mean smoothness  mean compactness  mean concavity  mean concave
points  \
count       569.000000        569.000000      569.000000
569.000000
mean          0.096360          0.104341        0.088799
0.048919
std           0.014064          0.052813        0.079720
0.038803
min           0.052630          0.019380        0.000000
0.000000
25%           0.086370          0.064920        0.029560
0.020310
50%           0.095870          0.092630        0.061540
0.033500
75%           0.105300          0.130400        0.130700
0.074000
max           0.163400          0.345400        0.426800
0.201200

       mean symmetry  mean fractal dimension  ...  worst texture  \
count     569.000000              569.000000  ...     569.000000
mean        0.181162                0.062798  ...      25.677223
std         0.027414                0.007060  ...       6.146258
min         0.106000                0.049960  ...      12.020000
25%         0.161900                0.057700  ...      21.080000
50%         0.179200                0.061540  ...      25.410000
75%         0.195700                0.066120  ...      29.720000
max         0.304000                0.097440  ...      49.540000

       worst perimeter   worst area  worst smoothness  worst
compactness  \
count       569.000000   569.000000        569.000000
569.000000
mean        107.261213   880.583128          0.132369
0.254265
std          33.602542   569.356993          0.022832
0.157336
min          50.410000   185.200000          0.071170
0.027290
25%          84.110000   515.300000          0.116600
```

```
0.147200
50%          97.660000   686.500000          0.131300
0.211900
75%         125.400000  1084.000000          0.146000
0.339100
max         251.200000  4254.000000          0.222600
1.058000

       worst concavity  worst concave points  worst symmetry  \
count       569.000000            569.000000      569.000000
mean          0.272188              0.114606        0.290076
std           0.208624              0.065732        0.061867
min           0.000000              0.000000        0.156500
25%           0.114500              0.064930        0.250400
50%           0.226700              0.099930        0.282200
75%           0.382900              0.161400        0.317900
max           1.252000              0.291000        0.663800

       worst fractal dimension      target
count               569.000000  569.000000
mean                  0.083946    0.627417
std                   0.018061    0.483918
min                   0.055040    0.000000
25%                   0.071460    0.000000
50%                   0.080040    1.000000
75%                   0.092080    1.000000
max                   0.207500    1.000000

[8 rows x 31 columns]
```

Check missing values in column

```
df_join.isnull().sum().sort_values(ascending=False)
```

```
mean radius                0
concavity error            0
worst fractal dimension    0
worst symmetry             0
worst concave points       0
worst concavity            0
worst compactness          0
worst smoothness           0
worst area                 0
worst perimeter            0
worst texture              0
worst radius               0
fractal dimension error    0
symmetry error             0
concave points error       0
compactness error          0
mean texture               0
```

```
smoothness error           0
area error                 0
perimeter error            0
texture error              0
radius error               0
mean fractal dimension     0
mean symmetry              0
mean concave points        0
mean concavity             0
mean compactness           0
mean smoothness            0
mean area                  0
mean perimeter             0
target                     0
dtype: int64
```

Count unique elements in each feature

```
df_join.nunique().sort_values(ascending=False)
```

```
smoothness error           547
fractal dimension error    545
worst area                 544
mean concave points        542
compactness error          541
radius error               540
worst concavity            539
mean area                  539
mean concavity             537
mean compactness           537
worst fractal dimension    535
concavity error            533
perimeter error            533
worst compactness          529
area error                 528
mean perimeter             522
texture error              519
worst perimeter            514
worst texture              511
concave points error       507
worst symmetry             500
mean fractal dimension     499
symmetry error             498
worst concave points       492
mean texture               479
mean smoothness            474
worst radius               457
mean radius                456
mean symmetry              432
worst smoothness           411
```

```
target                       2
dtype: int64
```

Utility

```
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, confusion_matrix

def evaluate_prediction(label_real, label_predict):
    accuration = accuracy_score(label_real, label_predict)
    precision = precision_score(label_real, label_predict)
    recall = recall_score(label_real, label_predict)
    f1 = f1_score(label_real, label_predict)
    confmatrix = confusion_matrix(label_real, label_predict)

    print(f"Accuracy score: {round(accuration, 2)}")
    print(f"Precision score: {round(precision, 2)}")
    print(f"Recall score: {round(recall, 2)}")
    print(f"F1 score: {round(f1, 2)}")
    print(f"Confusion matrix: {confmatrix}")
    return [round(accuration, 2), round(precision, 2), round(recall,
2), round(f1, 2)]
```

Algoritma: DecisionTreeClassifier

Melakukan pembelajaran

```
from sklearn import tree
clf = tree.DecisionTreeClassifier(random_state=10,
criterion="entropy")
clf = clf.fit(X_train, y_train)
tree.export_text(clf)
```

```
'|--- feature_22 <= 117.45\n|   |--- feature_27 <= 0.11\n|   |   |---
feature_3 <= 694.50\n|   |   |   |--- feature_10 <= 0.64\n|   |   |
|   |--- feature_21 <= 33.35\n|   |   |   |   |--- class: 1\n|   |
|   |   |--- feature_21 >  33.35\n|   |   |   |   |--- feature_22
<= 90.44\n|   |   |   |   |   |--- class: 1\n|   |   |   |   |
|--- feature_22 >  90.44\n|   |   |   |   |   |--- class: 0\n|   |
|   |--- feature_10 >  0.64\n|   |   |   |--- feature_10 <= 0.67\
n|   |   |   |   |   |--- class: 0\n|   |   |   |--- feature_10 >
0.67\n|   |   |   |   |--- class: 1\n|   |   |--- feature_3 >
694.50\n|   |   |--- feature_9 <= 0.06\n|   |   |   |---
class: 0\n|   |   |--- feature_9 >  0.06\n|   |   |   |---
class: 1\n|   |--- feature_27 >  0.11\n|   |   |--- feature_27 <=
0.16\n|   |   |--- feature_21 <= 27.48\n|   |   |   |---
feature_23 <= 811.10\n|   |   |   |   |--- class: 1\n|   |   |   |
|--- feature_23 >  811.10\n|   |   |   |   |--- feature_20 <=
16.62\n|   |   |   |   |   |--- feature_26 <= 0.26\n|   |   |   |
|   |   |--- class: 1\n|   |   |   |   |   |--- feature_26 >
0.26\n|   |   |   |   |   |--- class: 0\n|   |   |   |   |
|--- feature_20 >  16.62\n|   |   |   |   |--- class: 1\n|   |
```

```
|   |--- feature_21 >  27.48\n|   |   |   |   |--- feature_20 <=
15.22\n|   |   |   |   |   |--- feature_7 <= 0.05\n|   |   |   |   |
|   |--- class: 1\n|   |   |   |   |   |--- feature_7 >  0.05\n|   |
|   |   |   |   |--- class: 0\n|   |   |   |   |--- feature_20 >
15.22\n|   |   |   |   |   |--- feature_25 <= 0.33\n|   |   |   |   |
|   |--- class: 0\n|   |   |   |   |   |--- feature_25 >  0.33\n|   |
|   |   |   |--- feature_24 <= 0.14\n|   |   |   |   |   |   |
|--- class: 1\n|   |   |   |   |   |--- feature_24 >  0.14\n|   |
|   |   |   |   |   |--- class: 0\n|   |   |--- feature_27 >  0.16\n|
|   |   |--- feature_13 <= 13.72\n|   |   |   |   |--- class: 1\n|   |
|   |--- feature_13 >  13.72\n|   |   |   |   |--- class: 0\n|---
feature_22 >  117.45\n|   |--- class: 0\n'
```

Menyimpan hasil pembelajaran

```
import pickle

saved_model_tree = pickle.dumps(clf)
```

Melakukan proses prediksi dengan model yang telah disimpan

```
clf_load_tree = pickle.loads(saved_model_tree)
label_predict_tree = clf_load_tree.predict(X_test)
label_real = y_test.to_numpy()
```

Melakukan evaluasi hasil prediksi

```
res_dectree = evaluate_prediction(label_real, label_predict_tree)
```

```
Accuracy score: 0.92
Precision score: 0.99
Recall score: 0.89
F1 score: 0.94
Confusion matrix: [[38  1]
 [ 8 67]]
```

K-cross validation (k=10)

```
from sklearn.model_selection import cross_validate
from sklearn import linear_model
from sklearn.tree import DecisionTreeClassifier

cv_result = cross_validate(estimator=DecisionTreeClassifier(),
X=X_train, y=y_train, cv=10)
cv_result
```

```
{'fit_time': array([0.01599956, 0.02999687, 0.0138526 , 0.02809358,
0.01999664,
        0.02211952, 0.03734946, 0.02499819, 0.01801968, 0.02000403]),
 'score_time': array([0.00300121, 0.01401353, 0.00556898, 0.006001  ,
0.00882816,
        0.02153778, 0.00699592, 0.0060029 , 0.00351524, 0.00399566]),
 'test_score': array([0.93478261, 0.89130435, 0.95652174, 0.97826087,
0.91304348,
        0.91111111, 0.93333333, 0.91111111, 0.93333333, 0.88888889])}
```

Nilai-nilai adalah hasil dari *10-cross-validation*, yang membagi data menjadi 10 himpunan bagian, dan menggunakan 9 di antaranya untuk *training* dan 1 untuk *testing*, lalu mengulangi prosesnya sebanyak 10 kali.

Nilai `test_score` menunjukkan keakuratan model pada tiap set dari kesepuluh *subset*. Skor akurasi berkisar dari `0,89` hingga `0,98`, dengan skor rata-rata sekitar `0,93`. Hal ini menunjukkan bahwa model memiliki akurasi rata-rata 93% pada data yang tidak terlihat (*unseen data*).

Secara keseluruhan, hasil validasi *10-cross-validation* menunjukkan bahwa model memiliki performa yang baik dan dapat menggeneralisasi dengan baik pada data yang tidak terlihat mengingat skor akurasi yang konsisten pada tiap rangkaian tes yang berbeda.

Algoritma: Id3Estimator

Melakukan pembelajaran

```
# from id3 import Id3Estimator
# ! Library failed to used, have try to `pip install decision-tree-id3` and by cloning the project but still won't work
```

Menyimpan hasil pembelajaran

Melakukan proses prediksi dengan model yang telah disimpan

Melakukan evaluasi hasil prediksi

Algoritma: K-means

Melakukan pembelajaran

```
from sklearn.cluster import KMeans
kMeansModel = KMeans(n_clusters=2, n_init="auto",
random_state=10).fit(X_train)
kMeansModel
```

```
c:\Users\geral\anaconda3\lib\site-packages\sklearn\cluster\
_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on
Windows with MKL, when there are less chunks than available threads.
You can avoid it by setting the environment variable
OMP_NUM_THREADS=2.
  warnings.warn(
```

```
KMeans(n_clusters=2, n_init='auto', random_state=10)
```

Menyimpan hasil pembelajaran

```
saved_model_kmeans = pickle.dumps(kMeansModel)
```

Melakukan proses prediksi dengan model yang telah disimpan

```
kMeansModel_load = pickle.loads(saved_model_kmeans)
label_predict_kMeans = kMeansModel_load.fit_predict(X_test)
# Switching 0 to 1, and 1 to 0 since k-means only classify to two
classes without having interpretation to 0 and 1
label_predict_kMeans[label_predict_kMeans == 0] = 10 # Arbitrary
beholder number
label_predict_kMeans[label_predict_kMeans == 1] = 0
label_predict_kMeans[label_predict_kMeans == 10] = 1
```

```
c:\Users\geral\anaconda3\lib\site-packages\sklearn\cluster\
_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on
Windows with MKL, when there are less chunks than available threads.
You can avoid it by setting the environment variable
OMP_NUM_THREADS=1.
  warnings.warn(
```

Melakukan evaluasi hasil prediksi

```
res_kMeans = evaluate_prediction(label_real, label_predict_kMeans)
```

```
Accuracy score: 0.85
Precision score: 0.82
Recall score: 0.99
F1 score: 0.9
Confusion matrix: [[23 16]
 [ 1 74]]
```

Algoritma: LogisticRegression

Melakukan pembelajaran

```
from sklearn.linear_model import LogisticRegression
clf_lr = LogisticRegression(random_state=0,max_iter=2000).fit(X_train,
y_train)
```

```
c:\Users\geral\anaconda3\lib\site-packages\sklearn\linear_model\
_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

Menyimpan hasil pembelajaran

```
saved_model_lr = pickle.dumps(clf_lr)
```

Melakukan proses prediksi dengan model yang telah disimpan

```
clf_load_lr = pickle.loads(saved_model_lr)
label_predict_lr = clf_load_lr.predict(X_test)
```

Melakukan evaluasi hasil prediksi

```
res_lr = evaluate_prediction(label_real, label_predict_lr)
```

```
Accuracy score: 0.95
Precision score: 0.99
Recall score: 0.93
F1 score: 0.96
Confusion matrix: [[38  1]
 [ 5 70]]
```

Algoritma: Neural_network

Melakukan pembelajaran

```
from sklearn.neural_network import MLPClassifier
clf_nn = MLPClassifier(random_state=0, max_iter=300).fit(X_train,
y_train)
```

Menyimpan hasil pembelajaran

```
saved_model_nn = pickle.dumps(clf_nn)
```

Melakukan proses prediksi dengan model yang telah disimpan

```
clf_load_nn = pickle.loads(saved_model_nn)
label_predict_nn = clf_load_nn.predict(X_test)
```

Melakukan evaluasi hasil prediksi

```
res_nn = evaluate_prediction(label_real, label_predict_nn)
```

```
Accuracy score: 0.92
Precision score: 0.92
Recall score: 0.96
F1 score: 0.94
Confusion matrix: [[33  6]
 [ 3 72]]
```

Algoritma: SVM

Melakukan pembelajaran

```
from sklearn.svm import SVC
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
clf_svm = make_pipeline(StandardScaler(),
SVC(gamma='auto')).fit(X_train, y_train)
```

Menyimpan hasil pembelajaran

```
saved_model_svm = pickle.dumps(clf_svm)
```

Melakukan proses prediksi dengan model yang telah disimpan

```
clf_load_svm = pickle.loads(saved_model_svm)
label_predict_svm = clf_load_svm.predict(X_test)
```

Melakukan evaluasi hasil prediksi

```
res_svm = evaluate_prediction(label_real, label_predict_svm)
```

```
Accuracy score: 0.97
Precision score: 1.0
Recall score: 0.96
F1 score: 0.98
Confusion matrix: [[39  0]
 [ 3 72]]
```

```
print(res_dectree)
print(res_kMeans)
print(res_lr)
print(res_nn)
print(res_svm)
```

```
[0.92, 0.99, 0.89, 0.94]
[0.85, 0.82, 0.99, 0.9]
[0.95, 0.99, 0.93, 0.96]
[0.92, 0.92, 0.96, 0.94]
[0.97, 1.0, 0.96, 0.98]
```

Hasil analisis

- Data ini menunjukkan performa dari lima model yang berbeda pada *task* klasifikasi. Model *decision tree* mencapai skor akurasi 0,92, yang menunjukkan bahwa model tersebut mengklasifikasikan dengan benar 92% dari tiap instance pada tes set. Skor *precision*-nya adalah 0,99, artinya ketika memprediksi hasil yang positif, 99% hasilnya benar. Skor *recall* 0,89 menunjukkan bahwa model mengidentifikasi 89% kasus positif dengan benar. Skor F1 yang bernilai 0,94 menunjukkan ukuran trade-off antara *precision* dan *recall* dan merupakan *harmonic means* di antara keduanya.

- Model k-means mencapai skor akurasi 0,85, yang nilainya lebih rendah dari model *decision tree*. Skor *precision*-nya sebesar 0,82 yang juga lebih rendah dibandingkan *decision tree*, yang berarti hanya 82% kasus positif yang dapat diidentifikasi dengan benar. Namun, skor *recall*-nya adalah 0,99 tertinggi di antara semua model, yang berarti bahwa algoritma ini dapat mengidentifikasi hampir semua kasus positif dalam rangkaian tes. Skor F1-nya sebesar 0,9 lebih rendah dari semua model yang menunjukkan ketidakseimbangan skor *precision* dan *recall*

- Model *logistic regression* mencapai skor akurasi tertinggi kedua, yakni 0,95, yang menunjukkan bahwa model tersebut mengklasifikasikan dengan benar 95% *instance* dalam *test set*. Skor presisinya sebesar 0,99 yang besarnya sama dengan model *decision tree* dimana keduanya menunjukkan tingkat presisi yang tinggi. Skor recall sebesar 0,93 yang lebih rendah dari model *decision tree*, menunjukkan bahwa model tersebut gagal dalam memprediksi beberapa *instance*/contoh positif. Skor

F1-nya sebesar 0,96 yang merupakan skor F-1 tertinggi kedua di antara semua model yang menunjukkan keseimbangan yang baik antara skor *precision* dan *recall*.

- Model *neural network* mencapai skor akurasi 0,92 yang besarnya sama dengan model *decision tree*. Skor *precision*-nya sebesar 0,92 yang lebih rendah dari *decision tree* dan model *logistic regression*. Namun, skor *recall*-nya sebesar 0,96 yang lebih tinggi daripada model *decision tree* dan k-means yang menunjukkan bahwa model ini mengidentifikasi dengan tepat sebagian besar kasus positif. Skor F1 sebesar 0,94 yang sama besarnya dengan model *decision tree*.
- Model *support vector machine* (SVM) mencapai skor presisi tertinggi, yakni 1,0, yang menunjukkan bahwa model ini mengidentifikasi dengan benar semua *instance* positif dalam *test set*. Skor akurasinya yang sebesar 0,97 juga merupakan yang tertinggi di antara semua model. Skor *recall* 0,96 lebih rendah dari model *neural network* yang berarti model ini relatif lebih gagal dalam memprediksi beberapa *instance* positif. Skor F1-nya sebesar 0,98 yang merupakan nilai F-1 yang tertinggi di antara semua model, menunjukkan keseimbangan terbaik antara *precision* dan *recall*.

Secara keseluruhan, model *logistic regression* dan SVM merupakan model terbaik pada *task* klasifikasi pada kasus ini, dengan mencapai skor akurasi, *precision*, *recall*, dan F1 yang tinggi. *Decision tree* dan model *neural network* juga bekerja dengan baik, yakni mencapai skor akurasi dan F1 yang tinggi, tetapi dengan terdapat kompromi antara *precision* dan *recall*. Model k-means pada contoh kasus ini merupakan model yang terburuk dengan mencapai akurasi, precision, dan F1 yang lebih rendah dibandingkan keempat model lainnya.