

LAPORAN TUGAS BESAR
MATA KULIAH PEMBELAJARAN MESIN LANJUT

PENERAPAN TPOT DALAM AUTOMATED MACHINE LEARNING
PADA DATASET BREAST CANCER WISCONSIN (DIAGNOSTIC)

Diajukan Untuk Memenuhi Tugas Mata Kuliah Pembelajaran Mesin Lanjut

Dosen Pengampu: Isman Kurniawan, S.Pd., M.Si., M.Sc., Ph.D



Disusun Oleh:

Ibrahim Muhammad (1301194069)

Livia Naura Aqilla (1301194089)

PROGRAM STUDI S1 INFORMATIKA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM
BANDUNG
2022

Daftar Isi

1.	Formulasi Masalah	3
2.	Eksplorasi dan Persiapan Data.....	3
2.1	Eksplorasi data	3
2.2	Missing Value	5
2.3	Label Encoder	6
2.4	Drop Outliers.....	7
2.5	Splitting Data.....	8
3.	Pemodelan	8
4.	Eksperimen.....	9
5.	Evaluasi	10
6.	Kesimpulan.....	10

1. Formulasi Masalah

Tugas Besar ini bertujuan untuk membuat prediksi pada sebuah diagnosa terhadap pasien yang mungkin mengalami gejala penyakit Breast Cancer. Dataset yang digunakan pada tugas ini adalah dataset Breast Cancer Wisconsin. Penyelesaian masalah pada tugas ini menggunakan pemodelan Automated Machine Learning (auto ML). Pemodelan auto ML yang digunakan adalah Tree-Based Pipeline Optimization Tool (TPOT). Dalam tugas ini TPOT diaplikasikan untuk menemukan solusi terbaik dari permasalahan yang diselesaikan.

Agar mendapatkan prediksi yang optimal pada model yang digunakan, data terlebih dahulu diproses dalam tahap preprocessing. Setelah itu dilakukan cleaning dataset agar dapat diaplikasikan pada model auto ML yang digunakan. TPOT digunakan pada tugas yang dibangun karena bersifat open-source selain itu juga TPOT merupakan tools auto ML yang secara otomatis dapat menghasilkan model performa tinggi pada pemodelan sebuah data.

2. Eksplorasi dan Persiapan Data

Dataset yang digunakan adalah dataset Breast Cancer Wisconsin (Diagnostic). Eksplorasi dan persiapan data dilakukan dalam beberapa tahapan yang dijelaskan sebagai berikut

2.1 Eksplorasi data

Eksplorasi data dilakukan untuk memahami isi dan bentuk data sebelum masuk kedalam proses *pre-processing*

```
[ ] #import dataset
df = pd.read_csv('train.csv')
df.head(5)
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.1471	...
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	...
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.1279	...
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.1052	...
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.198	0.1043	...

5 rows x 32 columns

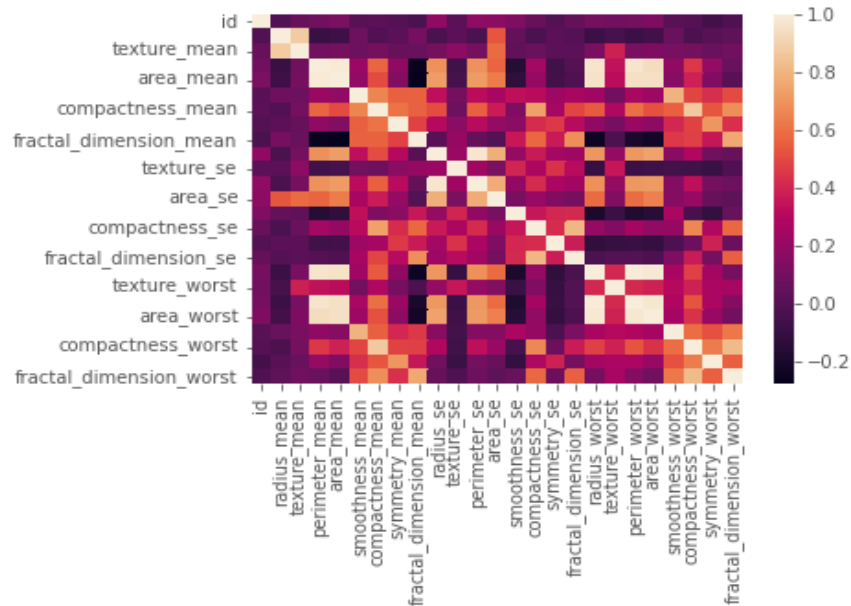
Gambar 1. Sample data yang digunakan

```
[ ] #info jumlah dan tipe data
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 463 entries, 0 to 462
Data columns (total 32 columns):
#   Column                                     Non-Null Count  Dtype
---  -
0   id                                         463 non-null    int64
1   diagnosis                                 463 non-null    object
2   radius_mean                              463 non-null    float64
3   texture_mean                             463 non-null    float64
4   perimeter_mean                           463 non-null    float64
5   area_mean                                463 non-null    float64
6   smoothness_mean                           463 non-null    float64
7   compactness_mean                          463 non-null    float64
8   concavity_mean                            463 non-null    object
9   concave points_mean                       463 non-null    object
10  symmetry_mean                             463 non-null    float64
11  fractal_dimension_mean                    463 non-null    float64
12  radius_se                                 463 non-null    float64
13  texture_se                                463 non-null    float64
14  perimeter_se                              463 non-null    float64
15  area_se                                   463 non-null    float64
16  smoothness_se                             463 non-null    float64
17  compactness_se                            463 non-null    float64
18  concavity_se                              463 non-null    object
19  concave points_se                         463 non-null    object
20  symmetry_se                               463 non-null    float64
21  fractal_dimension_se                      463 non-null    float64
22  radius_worst                              463 non-null    float64
23  texture_worst                             463 non-null    float64
24  perimeter_worst                           463 non-null    float64
25  area_worst                                463 non-null    float64
26  smoothness_worst                          463 non-null    float64
27  compactness_worst                         463 non-null    float64
28  concavity_worst                           463 non-null    object
29  concave points_worst                      463 non-null    object
30  symmetry_worst                             463 non-null    float64
31  fractal_dimension_worst                   463 non-null    float64
dtypes: float64(24), int64(1), object(7)
memory usage: 115.9+ KB
```

Gambar 2. Informasi tipe data tiap atribut dalam dataset.

Karena tidak ada data kosong dalam dataset, data numerik bisa langsung digambarkan kedalam heatmap yang berisi korelasi tiap atribut data masing-masing.



Gambar 3. Heatmap korelasi tiap data numerik

Dalam gambar 3 data yang berkorelasi positif yang bernilai mendekati 1 digambarkan dengan warna terang menuju putih, sedangkan data yang berkorelasi negatif yang bernilai kecil sampai negatif digambarkan dengan warna gelap menuju hitam.

2.2 Missing Value

Missing value dalam data dapat menurunkan bias dan akurasi model yang dibuat, maka keadaan *missing value* dalam data perlu dilihat dan dihapuskan jika ada. Dalam dataset tugas besar ini dapat dilihat tidak ada nilai kosong.

```
[ ] #untuk memeriksa kembali data kosong (missing value handling)
df.isna().sum()

id                0
diagnosis         0
radius_mean       0
texture_mean      0
perimeter_mean    0
area_mean         0
smoothness_mean   0
compactness_mean  0
concavity_mean    0
concave points_mean 0
symmetry_mean     0
fractal_dimension_mean 0
radius_se         0
texture_se        0
perimeter_se      0
area_se          0
smoothness_se     0
compactness_se    0
concavity_se      0
concave points_se 0
symmetry_se       0
fractal_dimension_se 0
radius_worst      0
texture_worst     0
perimeter_worst   0
area_worst        0
smoothness_worst  0
compactness_worst 0
concavity_worst   0
concave points_worst 0
symmetry_worst    0
fractal_dimension_worst 0
dtype: int64
```

Gambar 4. Pengecekan *missing value*

2.3 Label Encoder

Fitur encode dilakukan untuk mengubah tipe data yang non-numerik menjadi data tipe numerikal. Data yang diubah menjadi tipe data numerikal diantaranya adalah “diagnosis”, “concavity_mean”, “concave points_mean”, “concavity_se”, “concave points_se”, “concavity_worst”, dan “concave points_worst”.

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 463 entries, 0 to 462
Data columns (total 31 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   diagnosis                             463 non-null    int64
1   radius_mean                           463 non-null    float64
2   texture_mean                           463 non-null    float64
3   perimeter_mean                         463 non-null    float64
4   area_mean                             463 non-null    float64
5   smoothness_mean                        463 non-null    float64
6   compactness_mean                       463 non-null    float64
7   concavity_mean                         463 non-null    int64
8   concave points_mean                    463 non-null    int64
9   symmetry_mean                          463 non-null    float64
10  fractal_dimension_mean                 463 non-null    float64
11  radius_se                              463 non-null    float64
12  texture_se                              463 non-null    float64
13  perimeter_se                           463 non-null    float64
14  area_se                                463 non-null    float64
15  smoothness_se                          463 non-null    float64
16  compactness_se                         463 non-null    float64
17  concavity_se                           463 non-null    int64
18  concave points_se                      463 non-null    int64
19  symmetry_se                            463 non-null    float64
20  fractal_dimension_se                   463 non-null    float64
21  radius_worst                           463 non-null    float64
22  texture_worst                           463 non-null    float64
23  perimeter_worst                        463 non-null    float64
24  area_worst                             463 non-null    float64
25  smoothness_worst                       463 non-null    float64
26  compactness_worst                      463 non-null    float64
27  concavity_worst                        463 non-null    int64
28  concave points_worst                   463 non-null    int64

```

Gambar 5. Hasil encoder

2.4 Drop Outliers

Data outlier yaitu data yang nilainya menyimpang terlalu jauh dari data yang lainnya dalam suatu populasi data tertentu (pencilan). Outlier pada data mengakibatkan data menjadi bias dan tidak mencerminkan fenomena yang sebenarnya sehingga dapat menyebabkan performa dari model berkurang. Oleh karena itu data yang memiliki outlier harus dihilangkan. Hasil drop data outlier dapat dilihat pada gambar dibawah

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	...	radius_worst	texture_worst
1	1	20.570	17.77	132.90	1326.0	0.08474	0.07864	257	319	0.1812	...	24.99	23.41
2	1	19.690	21.25	130.00	1203.0	0.10960	0.15990	393	424	0.2069	...	23.57	25.53
4	1	20.290	14.34	135.10	1297.0	0.10030	0.13280	395	402	0.1809	...	22.54	16.67
6	1	18.250	19.98	119.60	1040.0	0.09463	0.10900	302	324	0.1794	...	22.88	27.66
7	1	13.710	20.83	90.20	577.9	0.11890	0.16450	269	290	0.2196	...	17.06	28.14
...
456	0	11.630	29.29	74.87	415.1	0.09357	0.08574	234	104	0.1799	...	13.12	38.81
457	0	13.210	25.25	84.10	537.9	0.08791	0.05205	102	110	0.1619	...	14.35	34.23
458	0	13.000	25.13	82.61	520.2	0.08369	0.05073	33	75	0.1667	...	14.34	31.88
459	0	9.755	28.20	61.68	290.9	0.07984	0.04626	45	32	0.1621	...	10.67	36.92
462	0	14.400	26.98	92.25	646.1	0.06995	0.05223	130	74	0.1707	...	15.40	31.98

Gambar 6. Hasil penghapusan data outlier

2.5 Splitting Data

Data split dilakukan untuk membagi data menjadi dua bagian. Bagian pertama digunakan untuk mengevaluasi data dan bagian kedua digunakan untuk melatih model yang dibangun. Pada penelitian ini data dibagi menjadi data train dan data test dengan porsi 75% untuk data train dan 25% untuk data test. Dengan data 'diagnosis' sebagai data target.

```
[184] #Split data dengan data target adalah fitur Diagnosis
      x_train, x_test, y_train, y_test = train_test_split(
          df.drop(['diagnosis'], axis=1), df['diagnosis'], test_size=0.2, random_state=42)

#Menampilkan panjang data hasil splitting
print(len(x_train), len(y_train))
print(len(x_test), len(y_test))

264 282
71 71
```

Gambar 7. Splitting data

3. Pemodelan

Dalam tugas besar ini digunakan Tree-based Pipeline Optimization Tool atau TPOT sebagai *Automation Tool* yang digunakan. Tahap pertama yang dilakukan adalah menetapkan parameter, parameter yang digunakan dalam generations bernilai 5, population_size bernilai 30, verbosity bernilai 2, random_state bernilai 40, cv bernilai 5 dan banyak jobs bernilai 1.

```
#memanggil function tpot
tpot = TPOTClassifier(
    generations=5, 5
    population_size=30,
    cv=5,
    random_state=40,
    verbosity=2,
    n_jobs=1)
```

Gambar 8. Parameter TPOT

Untuk persiapan data training dan testing dapat dilihat dalam gambar 9.

```
#training model
model = tpot.fit(x_train, y_train)
#prediksi data train
train_pred = model.predict(x_train)
#Evaluasi model untuk data train
train_acc = model.score(x_train, y_train)
train_f1 = f1_score(y_train, train_pred)
train_precision = precision_score(y_train, train_pred)
train_recall = recall_score(y_train, train_pred)

#prediksi data test
test_pred = model.predict(x_test)
#Evaluasi model untuk data test
test_acc = model.score(x_test, y_test)
test_f1 = f1_score(y_test, test_pred)
test_precision = precision_score(y_test, test_pred)
test_recall = recall_score(y_test, test_pred)
```

Gambar 9. Training dan testing model

4. Eksperimen

Pada tahap eksperimen, dilakukan perubahan terhadap parameter awal. Parameter yang diubah pada tahap ini yaitu generation yang awalnya di set 5 menjadi 10 ,dan ada penambahan konfigurasi operator yaitu TPOT Light. Konfigurasi TPOT Light digunakan untuk menemukan jalur pipa (pipeline) tercepat dan sederhana untuk permasalahan klasifikasi ataupun regresi. Dengan TPOT Light proses klasifikasi menjadi lebih cepat daripada sebelumnya. Hasil eksperimen dapat dilihat pada Gambar dibawah

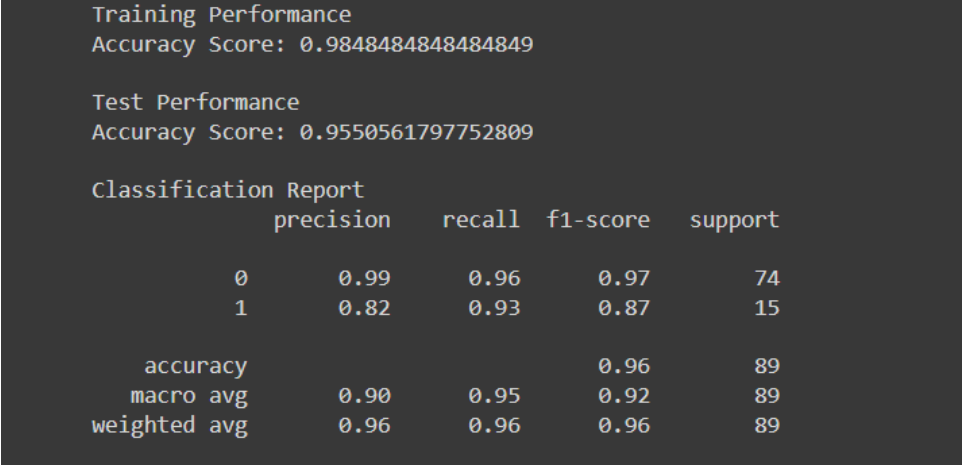
```
test_acc, test_recall, model, train_pred, test_pred = permodelan(x_train, x_test, y_train, y_test)

Generation 1 - Current best internal CV score: 0.958345428156749
Generation 2 - Current best internal CV score: 0.958345428156749
Generation 3 - Current best internal CV score: 0.9658925979680697
Generation 4 - Current best internal CV score: 0.9658925979680697
Generation 5 - Current best internal CV score: 0.9658925979680697
```

Gambar 10. Skor CV 5 generasi

5. Evaluasi

Pada tahap ini dilakukan evaluasi hasil dari eksperimen yang sudah dijalankan. Hasil evaluasi eksperimen dapat dilihat pada gambar 11.



```
Training Performance
Accuracy Score: 0.9848484848484849

Test Performance
Accuracy Score: 0.9550561797752809

Classification Report
precision    recall  f1-score   support

0           0.99      0.96      0.97         74
1           0.82      0.93      0.87         15

accuracy          0.96         89
macro avg         0.90      0.95      0.92         89
weighted avg      0.96      0.96      0.96         89
```

Gambar 11. Hasil eksperimen

Melalui tahap tahap eksplorasi dan preprocessing data, TPOT dapat menghasilkan model dengan akurasi hingga diatas 95%.

6. Kesimpulan

Automated Machine Learning (Auto ML) adalah proses otomasi terhadap tugas pada machine learning agar menjadi lebih cepat dan smart. Salah satu tools yang dapat digunakan untuk membuat program Auto ML adalah Tree-Based Pipeline Optimization Tool (TPOT). Untuk membuat pemodelan dengan menggunakan TPOT, harus melalui beberapa tahapan salah satunya tahap preprocessing data. Tahap ini dilakukan untuk membersihkan dan mengatur ulang struktur data agar dapat memberikan akurasi yang baik pada model auto ML yang diterapkan. Nilai akurasi yang dihasilkan dari model TPOT yang dibangun yaitu sebesar 95% yang didapatkan dari hasil eksperimen. Dari nilai tersebut dapat disimpulkan bahwa model TPOT yang dibangun telah mencapai nilai akurasi yang sangat baik karena berada diatas angka 90%