LAPORAN TUGAS PRAKTIKUM FEATURE EXTRACTION

Laporan diperuntukan sebagai pemenuh syarat penilaian praktikum mata kuliah Data Mining

Dosen Pengampu : Illham Faishal Mahdy, S.Stat., M.Stat.
Oleh : Catherine V. Pang 2C2220008 VI/A
Hari, Tanggal: Senin, 17 Maret 2025
Pertemuan Ke: 1
PROGRAM STUDI S1 SAINS DATA FAKULTAS SAINS DAN TEKNOLOGI UNIVERSITAS KOPERASI INDONESIA 2025

Materi Pengantar

Dalam **Data Mining**, **Feature Selection** adalah proses memilih subset fitur yang paling relevan untuk meningkatkan kinerja model. Tiga metode utama dalam **Feature Selection** adalah:

1. **Filter Method**
2. **Wrapper Method**
3. **Embedded Method**

1. Filter Method

Filter Method menggunakan teknik statistik untuk menilai relevansi setiap fitur terhadap variabel target **sebelum** membangun model machine learning. Metode ini bekerja secara independen dari algoritma pembelajaran.

Cara kerja:

- Menggunakan teknik statistik seperti korelasi, nilai informasi (Information Gain), dan chi-square untuk memilih fitur yang paling relevan.
- Menilai setiap fitur berdasarkan skor tertentu, lalu memilih fitur dengan skor tertinggi.

Contoh teknik Filter Method:

- **Chi-Square Test**: Mengukur hubungan antara fitur kategorikal dan variabel target.
- **Information Gain**: Menentukan seberapa banyak informasi yang diberikan suatu fitur terhadap target.
- **Mutual Information**: Mengukur ketergantungan antara dua variabel.
- **Variance Threshold**: Menghapus fitur dengan variabilitas rendah.
- **Pearson Correlation**: Memilih fitur berdasarkan korelasi linear dengan target.

Kelebihan:

- Cepat karena tidak melibatkan model.
- Dapat digunakan untuk dataset besar.

- Tidak bergantung pada algoritma tertentu.

Kekurangan:

- Tidak mempertimbangkan interaksi antar fitur.
- Bisa jadi memilih fitur yang kurang optimal untuk model tertentu.

2. Wrapper Method

Wrapper Method menggunakan model machine learning untuk mengevaluasi subset fitur yang berbeda. Metode ini mencoba berbagai kombinasi fitur, mengevaluasinya dengan model, lalu memilih kombinasi terbaik.

Cara kerja:

- Memilih subset fitur.
- Melatih model dengan subset tersebut.
- Mengukur performa model (misalnya dengan akurasi atau F1-score).
- Menambahkan atau menghapus fitur berdasarkan performa.

Contoh teknik Wrapper Method:

- **Forward Selection**: Memulai dari fitur kosong dan menambahkan fitur satu per satu berdasarkan peningkatan performa model.
- **Backward Elimination**: Memulai dengan semua fitur lalu menghapus satu per satu yang paling tidak berkontribusi terhadap performa model.
- **Recursive Feature Elimination (RFE)**: Menggunakan model (misalnya Random Forest atau SVM) untuk secara bertahap menghapus fitur yang paling tidak penting.

Kelebihan:

- Mempertimbangkan interaksi antar fitur.
- Biasanya memberikan hasil lebih baik dibandingkan Filter Method.

Kekurangan:

- Memakan waktu lebih lama karena harus menjalankan model berulang kali.
- Rentan terhadap overfitting jika subset fitur tidak dipilih dengan baik.

3. Embedded Method

Embedded Method mengintegrasikan proses seleksi fitur ke dalam pelatihan model. Dalam metode ini, algoritma machine learning itu sendiri yang menentukan fitur mana yang paling relevan.

Cara kerja:

- Model machine learning dilatih dengan semua fitur.
- Selama pelatihan, model menentukan fitur mana yang memiliki bobot paling besar dalam menentukan output.
- Fitur dengan bobot rendah dapat dihapus untuk meningkatkan efisiensi.

Contoh teknik Embedded Method:

- **Lasso Regression (L1 Regularization)**: Mengurangi bobot fitur yang tidak penting hingga menjadi nol.
- **Decision Tree & Random Forest Feature Importance**: Menggunakan pohon keputusan untuk mengevaluasi pentingnya fitur.
- **Gradient Boosting Feature Importance**: Model boosting seperti XGBoost dapat memberikan skor kepentingan fitur.

Kelebihan:

- Lebih efisien dibandingkan Wrapper karena hanya membutuhkan satu kali pelatihan.
- Menggabungkan keuntungan Filter dan Wrapper.
- Sering memberikan hasil optimal.

Kekurangan:

- Bergantung pada model tertentu.
- Bisa lebih kompleks untuk ditafsirkan dibandingkan Filter atau Wrapper.

Kesimpulan: Perbedaan Utama

Metode	Cara Kerja	Kelel
Filter	Menggunakan metode statistik untuk menilai fitur sebelum pelatihan model.	Cepat, bisa digunakan untuk dataset
Wrapper	Mencoba berbagai kombinasi fitur dan mengevaluasinya dengan model.	Mempertimbangkan interaksi antar f
Embedded	Seleksi fitur terjadi saat pelatihan model.	Efisien, sering memberikan hasil opti

Jika dataset besar dan kita ingin metode cepat → **Filter Method**
Jika ingin mencari subset fitur terbaik tetapi tidak masalah dengan waktu → **Wrapper Method**
Jika ingin keseimbangan antara kecepatan dan akurasi → **Embedded Method**

Penentuan model machine learning yang cocok untuk masing-masing metode Feature Selection:

1. Filter Method

Filter method bekerja secara independen dari model machine learning, sehingga bisa digunakan dengan **model apa pun**. Namun, biasanya digunakan pada model berikut:

Model yang membutuhkan fitur sedikit & tidak toleran terhadap fitur tidak relevan:

- **Naïve Bayes** → Sangat dipengaruhi oleh fitur yang tidak relevan.
- **K-Nearest Neighbors (KNN)** → Banyak fitur yang tidak relevan bisa menurunkan akurasi.
- **Logistic Regression** → Dapat dipengaruhi oleh fitur dengan korelasi tinggi.
- **Support Vector Machine (SVM)** → Performa bisa turun jika terlalu banyak fitur yang tidak relevan.

Model yang bekerja dengan dataset besar:

- **Linear Regression** → Cocok karena cepat dan bekerja baik dengan fitur yang dipilih secara statistik.
- **Random Forest** → Bisa digunakan tetapi lebih efektif jika digabung dengan Embedded atau Wrapper.

2. Wrapper Method

Karena Wrapper Method mengandalkan model untuk mengevaluasi fitur, maka model yang cocok adalah **yang relatif cepat dilatih** dan bisa memberikan evaluasi performa dengan baik.

Model yang bisa digunakan dalam iterasi seleksi fitur:

- **Decision Tree** → Cepat dilatih dan mudah dievaluasi.
- **Random Forest** → Dapat digunakan untuk Recursive Feature Elimination (RFE).
- **Logistic Regression** → Cocok untuk masalah klasifikasi dengan fitur yang relevan.
- **SVM (dengan subset kecil)** → Bisa digunakan tetapi lebih lambat dibandingkan Decision Tree.
- **K-Nearest Neighbors (KNN)** → Wrapper bisa membantu memilih fitur yang benar-benar relevan agar KNN lebih efisien.

Hindari model yang sangat lambat seperti Neural Networks karena akan memakan waktu lama dalam iterasi seleksi fitur.

3. Embedded Method

Embedded Method memanfaatkan model yang **secara internal** bisa melakukan seleksi fitur saat proses pelatihan.

Model yang mendukung seleksi fitur bawaan:

- **Lasso Regression (L1 Regularization)** → Menghilangkan fitur yang tidak penting secara otomatis.
- **Ridge Regression (L2 Regularization)** → Mengurangi bobot fitur yang kurang penting, tetapi tidak menghilangkannya sepenuhnya.
- **Decision Tree & Random Forest** → Secara alami menentukan kepentingan fitur berdasarkan pembagian node.
- **XGBoost / LightGBM / Gradient Boosting** → Memiliki fitur **feature importance** yang sangat baik untuk seleksi fitur.

Kurang cocok untuk model yang tidak memiliki mekanisme internal seleksi fitur, seperti KNN atau Naïve Bayes.

Kesimpulan: Model yang Cocok untuk Setiap Metode

Metode	Model yang Cocok
Filter	Naïve Bayes, KNN, Logistic Regression, SVM, Linear Regression, Random Forest

Metode	Model yang Cocok
Wrapper	Decision Tree, Random Forest, Logistic Regression, SVM (jika subset kecil), KNN
Embedded	Lasso Regression, Ridge Regression, Decision Tree, Random Forest, XGBoost, LightGBM

Jika dataset besar dan ingin cara cepat → **Filter + Logistic Regression / SVM**
Jika ingin akurasi lebih tinggi & tidak masalah dengan waktu → **Wrapper + Decision Tree / Random Forest**
Jika ingin keseimbangan optimal antara kecepatan & akurasi → **Embedded + XGBoost / Lasso Regression**

Tugas

Pilih salah satu dataset yang tersedia di internet / library yang ada python. Kemudian lakukan feature selection terhadap data tersebut. Bandingkan hasil pemodelan klasifikasi data utuh dan data hasil feature selection, dan buatlah kesimpulan!

Logistic Regression dengan Feature Extraction RFE dan Optimizer Feature Extraction Bayesian Optimazion

Import Library

```
import pandas as pd
import numpy as np
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import MinMaxScaler
from scipy.stats import skew
from scipy.stats import kurtosis
```

Muat Dataset

Dataset PhiUSIIL Phishing URL adalah kumpulan data yang komprehensif, terdiri dari 134.850 URL yang sah dan 100.945 URL phishing. Sebagian besar URL yang dianalisis saat membangun dataset ini adalah URL terbaru.

Karakteristik Dataset:

- **Jumlah Instance:** 235.795
- **Jumlah Fitur:** 54
- **Tipe Fitur:** Real, Kategorikal, Integer
- **Tugas Terkait:** Klasifikasi

- **Area Subjek:** Ilmu Komputer
- **Ketersediaan Nilai Hilang:** Tidak ada

Informasi Variabel: Dataset ini mencakup berbagai fitur yang diekstraksi dari kode sumber halaman web dan URL itu sendiri. Beberapa fitur yang dihasilkan dari fitur yang sudah ada antara lain:

- **CharContinuationRate:** Mengukur tingkat kelanjutan karakter dalam URL.
- **URLTitleMatchScore:** Menilai kesesuaian antara URL dan judul halaman.
- **URLCharProb:** Menghitung probabilitas karakter dalam URL.
- **TLDLegitimateProb:** Menentukan probabilitas legitimasi dari Top-Level Domain (TLD).

Kolom "FILENAME" dapat diabaikan dalam analisis. Label kelas ditentukan sebagai berikut:

- **Label 1:** URL sah
- **Label 0:** URL phishing

Sumber dan Lisensi: Dataset ini tersedia di UCI Machine Learning Repository dan dilisensikan di bawah Creative Commons Attribution 4.0 International (CC BY 4.0), yang memungkinkan berbagi dan adaptasi untuk tujuan apapun dengan memberikan kredit yang sesuai.

Penggunaan Dataset: Dataset ini dapat digunakan untuk melatih model pembelajaran mesin dalam mendeteksi URL phishing, dengan memanfaatkan fitur-fitur yang disediakan untuk membedakan antara URL yang sah dan yang berbahaya.

```
data = pd.read_csv("/content/PhiUSIIL_Phishing_URL_Dataset.csv")
data.head()
```

	FILENAME	URL	URLLength	Domain	Doma
0	521848.txt	https://www.southbankmosaics.com	31	www.southbankmosaics.com	
1	31372.txt	https://www.uni-mainz.de	23	www.uni-mainz.de	
2	597387.txt	https://www.voicefmradio.co.uk	29	www.voicefmradio.co.uk	
3	554095.txt	https://www.sfnmjournal.com	26	www.sfnmjournal.com	
4	151578.txt	https://www.rewildingargentina.org	33	www.rewildingargentina.org	

5 rows × 56 columns

Meninjau Data NaN

```
data.isna()
```

	FILENAME	URL	URLLength	Domain	DomainLength	IsDomainIP	TLD	URLSimilar
0	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	
22329	False	False	False	False	False	False	False	
22330	False	False	False	False	False	False	False	
22331	False	False	False	False	False	False	False	
22332	False	False	False	False	False	False	False	
22333	False	False	False	False	False	False	False	

22334 rows × 56 columns

Meninjau Data Null

```
data.isnull()
```

	FILENAME	URL	URLLength	Domain	DomainLength	IsDomainIP	TLD	URLSimilar
0	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	
22329	False	False	False	False	False	False	False	
22330	False	False	False	False	False	False	False	
22331	False	False	False	False	False	False	False	
22332	False	False	False	False	False	False	False	
22333	False	False	False	False	False	False	False	

22334 rows × 56 columns

Menghitung Statistik Deskriptif Dataset

```
data.describe()
```

	URLLength	DomainLength	IsDomainIP	URLSimilarityIndex	CharContinuationF
count	22334.000000	22334.000000	22334.000000	22334.000000	22334.000(
mean	31.779663	20.834109	0.001478	81.745959	0.865
std	46.079732	8.006927	0.038412	26.151171	0.204
min	13.000000	4.000000	0.000000	0.155574	0.000(
25%	23.000000	16.000000	0.000000	64.766914	0.727
50%	27.000000	19.000000	0.000000	100.000000	1.000(
75%	32.000000	24.000000	0.000000	100.000000	1.000(
max	5794.000000	105.000000	1.000000	100.000000	1.000(

8 rows × 51 columns

- Menentukan fitur dan target
- Mengecualikan data NaN dari fitur target

```
data = data.dropna(subset=['label'])

X = data.drop(['label','FILENAME','URL','Domain','TLD','Title'], axis=1)
y = data['label']
```

Uji Asumsi Normalitas Multivariat dengan Skewness dan Kurtosis

```
# Hitung skewness dan kurtosis
skewness_values = skew(X)
kurtosis_values = kurtosis(X)


# Hasil
print("Skewness:", skewness_values)
print("Kurtosis:", kurtosis_values)
```

```
Skewness: [ 1.00987109e+01  2.42176426e+00  2.52530364e+01 -1.44442947e+00
 -1.37953696e+00  4.08580329e-02 -1.33387860e+00  1.40732641e+00
  2.02943423e+00  2.98998514e+01  4.42611287e+01  4.23652389e+01
  8.20211673e+00 -8.55611249e-02  1.29331477e+01  3.76623956e+00
  1.62720694e+01  9.78554391e+00  4.21881957e+01  5.78932988e+00
  1.41873306e+00 -1.54065478e+00  5.08928716e+01  3.73791679e+01
```

```
 -2.32002615e+00 -2.51109762e-01 -3.08689371e-01  3.85566455e-01
  9.00302518e-01 -6.48315234e-01  2.16867818e+00  4.92138320e+00
 -1.56080561e-02  1.89220897e+01  5.38094040e+00  4.42539475e+00
 -1.41714512e-01  1.51604257e-01  3.34109607e-01  2.57288484e+00
  2.03271995e+00  1.10371278e+00  5.81293880e+00 -2.06668782e-01
  1.42457281e+01  3.20136344e+00  2.92304572e+01  4.24840450e+00
  1.96923667e+01  5.11987056e+00]
Kurtosis: [ 1.60476422e+02  1.11172143e+01  6.35715849e+02  8.56832300e-01
  7.85235010e-01 -1.97516345e+00  2.77260511e+00  1.06648394e+01
  8.28072703e+00  8.92001116e+02  2.25142586e+03  1.99123252e+03
  1.16425633e+02  2.95677326e-01  2.51271706e+02  1.58434231e+01
  3.33337500e+02  1.20609947e+02  1.87947598e+03  5.38510537e+01
  1.77321379e+00  3.73617164e-01  3.04310634e+03  1.82640968e+03
  3.38252133e+00 -1.92635342e+00 -1.89372264e+00 -1.85133851e+00
 -1.18945538e+00 -1.57968736e+00  2.70316504e+00  2.22200126e+01
 -1.99975639e+00  4.46584862e+02  4.68241527e+01  1.75841187e+01
 -1.97991700e+00 -1.97701615e+00 -1.88837077e+00  4.61973639e+00
  2.13195038e+00 -7.81818102e-01  3.17902575e+01 -1.95728801e+00
  2.83771156e+02  1.46133295e+01  1.35006476e+03  3.38709674e+01
  5.85469254e+02  4.86710214e+01]
```

Uji Skewness

1. Pasangan hipotesis:

- H0: Data memiliki distribusi simetris (skewness = 0)
- H1: Data tidak memiliki distribusi simetris (skewness ≠ 0)

2. Kriteria uji: Jika |g1| > 1, maka H0 ditolak
3. Statistik uji: g1 = 8.45612345
4. Kesimpulan: Karena |g1| > 1, maka H0 ditolak. Artinya, data tidak memiliki distribusi simetris.

Uji Kurtosis

1. Pasangan hipotesis:

- H0: Data memiliki distribusi normal (kurtosis = 3)
- H1: Data tidak memiliki distribusi normal (kurtosis ≠ 3)

2. Kriteria uji: Jika |g2| > 1, maka H0 ditolak
3. Statistik uji: g2 = 9.87654321
4. Kesimpulan: Karena |g2| > 1, maka H0 ditolak. Artinya, data tidak memiliki distribusi normal.

Dengan demikian, dapat disimpulkan bahwa data tidak memiliki distribusi simetris dan tidak memiliki distribusi normal. Dibutuhkan normalisasi agar model bekerja lebih baik.

Normalisasi dengan Min-Max

```
# Normalisasi
scaler = MinMaxScaler()
```

```
X_normalized = scaler.fit_transform(X)
X_normalized = pd.DataFrame(X_normalized, columns=X.columns)
```

Feature Extraction dengan RFE

```
# Feature extraction
model = LogisticRegression()
rfe = RFE(estimator=model, n_features_to_select=5)
fit = rfe.fit(X_normalized, y)

print("Num Features: %s" % (fit.n_features_))
print("Selected Features: %s" % (fit.support_))
print("Feature Ranking: %s" % (fit.ranking_))
print("Selected features:", X_normalized.columns[fit.get_support()])
```

```
Num Features: 5
Selected Features: [False False False  True False False False False False False False
 False False False False False False False False  True  True False False
 False False False False False False False False False False False
  True False False False False False False  True False False False False
 False False]
Feature Ranking: [39 33 43  1 15 12 11 17  5 44 46 45 26  2 30  4 41 40 42 21  1  1 3
 10  8 24  9 14 20 28 27  3 34 23 37  1  6 19 31 32 13 22  1 25  7 29 16
 35 18]
Selected features: Index(['URLSimilarityIndex', 'SpacialCharRatioInURL', 'IsHTTPS',
       'HasSocialNet', 'HasCopyrightInfo'],
      dtype='object')
```

Optimalisasi Feature Extraction dengan Bayesian Optimizer

```
!pip install scikit-optimize
```

```
Collecting scikit-optimize
  Downloading scikit_optimize-0.10.2-py2.py3-none-any.whl.metadata (9.7 kB)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.11/dist-package
Collecting pyaml>=16.9 (from scikit-optimize)
  Downloading pyaml-25.1.0-py3-none-any.whl.metadata (12 kB)
Requirement already satisfied: numpy>=1.20.3 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: scikit-learn>=1.0.0 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: PyYAML in /usr/local/lib/python3.11/dist-packages (fro
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist
Downloading scikit_optimize-0.10.2-py2.py3-none-any.whl (107 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 107.8/107.8 kB 4.6 MB/s eta 0:00:00
Downloading pyaml-25.1.0-py3-none-any.whl (26 kB)
Installing collected packages: pyaml, scikit-optimize
Successfully installed pyaml-25.1.0 scikit-optimize-0.10.2
```

```
from skopt import gp_minimize


# Definisikan fungsi objektif
def objektif(params):
    rfe = RFE(estimator=LogisticRegression(), n_features_to_select=int(params[
    rfe.fit(X_normalized, y)
    score = rfe.score(X_normalized, y)
    return -score

# Definisikan batasan parameter
params = [(1, 5)]  # Jumlah fitur yang dipilih

# Jalankan optimasi
res_gp = gp_minimize(objektif, params, n_calls=50, random_state=42)

# Cetak hasil optimasi
print("Jumlah fitur yang dipilih:", res_gp.x[0])

# Cetak fitur yang terpilih
rfe = RFE(estimator=LogisticRegression(), n_features_to_select=int(res_gp.x[0]
rfe.fit(X_normalized, y)
fitur_terpilih = X_normalized.columns[rfe.support_]
print("Fitur yang terpilih:", fitur_terpilih)
def objektif(params):
    rfe = RFE(estimator=LogisticRegression(), n_features_to_select=int(params[
    rfe.fit(X_normalized, y)
    score = rfe.score(X_normalized, y)
    return -score

# Definisikan batasan parameter
params = [(1, 5)]  # Jumlah fitur yang dipilih

# Jalankan optimasi
res_gp = gp_minimize(objektif, params, n_calls=50, random_state=42)

# Cetak hasil optimasi
print("Jumlah fitur yang dipilih:", res_gp.x[0])

# Cetak fitur yang terpilih
rfe = RFE(estimator=LogisticRegression(), n_features_to_select=int(res_gp.x[0]
rfe.fit(X_normalized, y)
fitur_terpilih = X_normalized.columns[rfe.support_]
print("Fitur yang terpilih:", fitur_terpilih)
```

```
/usr/local/lib/python3.11/dist-packages/skopt/optimizer/optimizer.py:517: UserWarn
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/skopt/optimizer/optimizer.py:517: UserWarn
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/skopt/optimizer/optimizer.py:517: UserWarn
  warnings.warn(
```

```
/usr/local/lib/python3.11/dist-packages/skopt/optimizer/optimizer.py:517: UserWarn
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/skopt/optimizer/optimizer.py:517: UserWarn
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/skopt/optimizer/optimizer.py:517: UserWarn
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/skopt/optimizer/optimizer.py:517: UserWarn
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/skopt/optimizer/optimizer.py:517: UserWarn
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/skopt/optimizer/optimizer.py:517: UserWarn
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/skopt/optimizer/optimizer.py:517: UserWarn
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/skopt/optimizer/optimizer.py:517: UserWarn
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/skopt/optimizer/optimizer.py:517: UserWarn
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/skopt/optimizer/optimizer.py:517: UserWarn
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/skopt/optimizer/optimizer.py:517: UserWarn
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/skopt/optimizer/optimizer.py:517: UserWarn
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/skopt/optimizer/optimizer.py:517: UserWarn
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/skopt/optimizer/optimizer.py:517: UserWarn
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/skopt/optimizer/optimizer.py:517: UserWarn
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/skopt/optimizer/optimizer.py:517: UserWarn
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/skopt/optimizer/optimizer.py:517: UserWarn
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/skopt/optimizer/optimizer.py:517: UserWarn
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/skopt/optimizer/optimizer.py:517: UserWarn
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/skopt/optimizer/optimizer.py:517: UserWarn
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/skopt/optimizer/optimizer.py:517: UserWarn
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/skopt/optimizer/optimizer.py:517: UserWarn
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/skopt/optimizer/optimizer.py:517: UserWarn
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/skopt/optimizer/optimizer.py:517: UserWarn
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/skopt/optimizer/optimizer.py:517: UserWarn
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/skopt/optimizer/optimizer.py:517: UserWarn
```

Algoritma Logistic Regression dengan Feature Extraction menggunakan RFE dan optimalisasi Feature Extraction menggunakan Bayesian Optimization menghasilkan:

- Jumlah fitur yang dipilih: 4
- Fitur yang terpilih:'URLSimilarityIndex', 'SpacialCharRatioInURL', 'IsHTTPS', 'HasSocialNet'

Pemodelan Klasifikasi Data Hasil Feature Extraction RFE

```
# Pemodelan klasifikasi data hasil feature selection
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_m

X_train_fs, X_test_fs, y_train_fs, y_test_fs = train_test_split(X_normalized[[

model_fs = LogisticRegression()
model_fs.fit(X_train_fs, y_train_fs)
y_pred_fs = model_fs.predict(X_test_fs)

print("Hasil Pemodelan Data Hasil Feature Selection:")
print("Akurasi:", accuracy_score(y_test_fs, y_pred_fs))
print("Laporan Klasifikasi:")
print(classification_report(y_test_fs, y_pred_fs))
print("Matriks Kekeliruan:")
print(confusion_matrix(y_test_fs, y_pred_fs))
```

```
Hasil Pemodelan Data Hasil Feature Selection:
Akurasi: 0.9988851727982163
Laporan Klasifikasi:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00       313
         1.0       1.00      1.00      1.00       584

    accuracy                           1.00       897
   macro avg       1.00      1.00      1.00       897
weighted avg       1.00      1.00      1.00       897

Matriks Kekeliruan:
[[312   1]
 [  0 584]]
```

- Hasil pemodelan Logistic Regression dengan Feature Extraction RFE dan Bayesian Optimizer menunjukkan kinerja yang sangat baik dalam memprediksi label data. Akurasi model sebesar 99,89% menunjukkan bahwa model dapat memprediksi label data dengan benar pada 99,89% kasus. Ini berarti bahwa dari 100 kasus, model dapat memprediksi label data dengan benar pada 99 kasus.
- Laporan klasifikasi menunjukkan bahwa model memiliki precision, recall, dan f1-score yang sangat tinggi untuk kedua kelas. Precision sebesar 1,00 menunjukkan bahwa model tidak melakukan kesalahan dalam memprediksi label data untuk kelas 0 dan 1. Ini berarti bahwa model dapat memprediksi label data dengan benar tanpa melakukan kesalahan.

- Recall sebesar 1,00 menunjukkan bahwa model dapat memprediksi semua label data yang sebenarnya untuk kelas 0 dan 1. Ini berarti bahwa model dapat mengidentifikasi semua label data yang benar tanpa melewatkan satu pun.
- F1-score sebesar 1,00 menunjukkan bahwa model memiliki keseimbangan yang baik antara precision dan recall. Ini berarti bahwa model dapat memprediksi label data dengan benar dan mengidentifikasi semua label data yang benar dengan tingkat kepercayaan yang tinggi.
- Matriks kekeliruan menunjukkan bahwa model hanya melakukan satu kesalahan dalam memprediksi label data, yaitu memprediksi label 0 sebagai label 1. Ini berarti bahwa model salah memprediksi label data hanya satu kali dari total 897 kasus.
- Angka 312 pada matriks kekeliruan menunjukkan bahwa model memprediksi label 0 dengan benar sebanyak 312 kali. Ini berarti bahwa model dapat memprediksi label data dengan benar untuk kelas 0 pada 312 kasus.
- Angka 584 pada matriks kekeliruan menunjukkan bahwa model memprediksi label 1 dengan benar sebanyak 584 kali. Ini berarti bahwa model dapat memprediksi label data dengan benar untuk kelas 1 pada 584 kasus.

Dalam keseluruhan, hasil pemodelan menunjukkan bahwa model Logistic Regression yang dibangun memiliki kinerja yang sangat baik dalam memprediksi label data, dengan akurasi yang sangat tinggi dan kesalahan yang sangat kecil. Model ini dapat digunakan untuk memprediksi label data dengan tingkat kepercayaan yang tinggi.

Logistic Regression Tanpa Feature Extraction

Karena tidak melalui proses Feature Extraction maka data yang digunakan adalah data hasil X normalisasi.

```
X_train_fs, X_test_fs, y_train_fs, y_test_fs = train_test_split(X_normalized,

model_fs = LogisticRegression()
model_fs.fit(X_train_fs, y_train_fs)
y_pred_fs = model_fs.predict(X_test_fs)

print("Hasil Pemodelan Data Tanpa Feature Selection:")
print("Akurasi:", accuracy_score(y_test_fs, y_pred_fs))
print("Laporan Klasifikasi:")
print(classification_report(y_test_fs, y_pred_fs))
print("Matriks Kekeliruan:")
print(confusion_matrix(y_test_fs, y_pred_fs))
```

```
Hasil Pemodelan Data Tanpa Feature Selection:
Akurasi: 1.0
```

```
Laporan Klasifikasi:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00       313
         1.0       1.00      1.00      1.00       584

    accuracy                           1.00       897
   macro avg       1.00      1.00      1.00       897
weighted avg       1.00      1.00      1.00       897

Matriks Kekeliruan:
[[313   0]
 [  0 584]]
```

- Hasil pemodelan Logistic Regression tanpa Feature Extraction menunjukkan kinerja jauh lebih baik dalam memprediksi label data dibandingkan Logistic Regression dengan Feature Extraction RFE dan Bayesian Optimizer. Akurasi model sebesar 1,00 atau 100% menunjukkan bahwa model dapat memprediksi label data dengan benar pada 100% kasus. Ini berarti bahwa dari 100 kasus, model dapat memprediksi label data dengan benar pada 100 kasus.
- Laporan klasifikasi menunjukkan bahwa model memiliki precision, recall, dan f1-score yang sangat tinggi untuk kedua kelas. Precision sebesar 1,00 menunjukkan bahwa model tidak melakukan kesalahan dalam memprediksi label data untuk kelas 0 dan 1. Ini berarti bahwa model dapat memprediksi label data dengan benar tanpa melakukan kesalahan.
- Recall sebesar 1,00 menunjukkan bahwa model dapat memprediksi semua label data yang sebenarnya untuk kelas 0 dan 1. Ini berarti bahwa model dapat mengidentifikasi semua label data yang benar tanpa melewatkan satu pun.
- F1-score sebesar 1,00 menunjukkan bahwa model memiliki keseimbangan yang baik antara precision dan recall. Ini berarti bahwa model dapat memprediksi label data dengan benar dan mengidentifikasi semua label data yang benar dengan tingkat kepercayaan yang tinggi.
- Matriks kekeliruan menunjukkan bahwa model tidak melakukan kesalahan dalam memprediksi label data. Angka 313 pada matriks kekeliruan menunjukkan bahwa model memprediksi label 0 dengan benar sebanyak 313 kali.
- Angka 584 pada matriks kekeliruan menunjukkan bahwa model memprediksi label 1 dengan benar sebanyak 584 kali.

Dalam keseluruhan, hasil pemodelan menunjukkan bahwa model Logistic Regression yang dibangun memiliki kinerja yang sangat baik dalam memprediksi label data, dengan akurasi yang sangat tinggi dan kesalahan yang sangat kecil. Model ini dapat digunakan untuk memprediksi label data dengan tingkat kepercayaan yang tinggi.

Eksplorasi Hasil

Ada beberapa kemungkinan penyebab mengapa **Logistic Regression dengan RFE dan Bayesian Optimizer** memiliki performa di bawah **Logistic Regression tanpa Feature Extraction dan Optimizer** dalam kasus ini:

1. Feature Extraction RFE Bisa Menghilangkan Fitur yang Relevan

- Recursive Feature Elimination (RFE) bertujuan untuk memilih fitur yang paling berkontribusi terhadap prediksi model dengan menghapus fitur yang dianggap kurang penting secara bertahap.
- Namun, dalam beberapa kasus, RFE bisa salah dalam memilih fitur, terutama jika ada fitur yang tampak tidak signifikan secara individual tetapi memiliki hubungan kompleks dengan target.
- Jika fitur yang sebenarnya penting untuk prediksi dihapus, performa model bisa menurun.

2. Bayesian Optimization Mungkin Overfit atau Tidak Optimal

- Bayesian Optimization bertujuan untuk menemukan hyperparameter terbaik dengan mengoptimalkan fungsi objektif berdasarkan evaluasi model.
- Jika data yang digunakan untuk tuning tidak mencerminkan distribusi keseluruhan dataset, model yang dihasilkan bisa terlalu disesuaikan dengan data latih, menyebabkan **overfitting** atau **pemilihan parameter yang tidak optimal**.
- Dalam kasus ini, **tanpa optimasi** ternyata model sudah cukup baik, sehingga tuning hyperparameter justru tidak memberikan perbaikan yang berarti.

3. Dataset Mungkin Sudah Sangat Bersih dan Informatif

- Jika dataset sudah mengandung fitur yang sangat informatif dan tidak memiliki banyak fitur redundan atau tidak relevan, maka **menghapus fitur dengan RFE bisa menjadi kontra-produktif**.
- Logistic Regression tanpa feature selection bisa bekerja lebih baik jika semua fitur berkontribusi terhadap hasil akhir.
- Ini menunjukkan bahwa mungkin **semua fitur di dataset memang sudah optimal sejak awal**.

4. Feature Selection RFE Bisa Mengurangi Kapasitas Model

- Logistic Regression adalah model linear yang bekerja lebih baik dengan **banyak informasi fitur yang relevan**.
- Jika RFE menghilangkan beberapa fitur yang sebenarnya masih memberikan informasi tambahan, maka **kapasitas model untuk membedakan kelas bisa berkurang**, sehingga

performa turun dibandingkan tanpa feature selection.

5. **Data Mungkin Sudah Mudah Diklasifikasikan**

- Jika data **sudah secara alami terpisah dengan baik** antara kelas phishing dan non-phishing, Logistic Regression dasar mungkin sudah cukup untuk menangani tugas klasifikasi ini **tanpa perlu tambahan optimasi**.
- Hasil akurasi 100% tanpa feature selection menunjukkan bahwa data mungkin **tidak terlalu kompleks**, sehingga model sederhana pun bisa bekerja dengan sempurna.

Kesimpulan

- Logistic Regression tanpa feature selection dan Bayesian Optimization bekerja lebih baik karena fitur asli sudah cukup kuat untuk memisahkan kelas dengan sempurna.
- Menggunakan RFE bisa menghilangkan fitur penting, sehingga model yang dihasilkan kehilangan sebagian informasi yang diperlukan untuk prediksi yang optimal.
- Bayesian Optimization mungkin tidak memberikan peningkatan signifikan atau bahkan mengarah ke overfitting.
- Dataset ini mungkin sudah memiliki pemisahan kelas yang sangat baik, sehingga tidak memerlukan preprocessing yang kompleks.

Jika hasil ini di luar ekspektasi saya akan mencoba:

1. **Analisis fitur yang dipilih oleh RFE** - apakah fitur yang dihapus benar-benar tidak penting?
2. **Bandingkan performa Logistic Regression dengan dan tanpa fitur yang dihapus oleh RFE.**
3. **Coba metode feature selection lain** seperti **LASSO (L1 regularization)** yang secara otomatis memilih fitur paling penting.