



DETEKSI HALAMAN WEBSITE PHISHING

MENGGUNAKAN ALGORITMA MACHINE LEARNING GRADIENT BOOSTING CLASSIFIER

Muhammad Rizal Supriadi

Roni Andarsyah

**DETEKSI HALAMAN WEBSITE
PHISHING MENGGUNAKAN
ALGORITMA MACHINE LEARNING
GRADIENT BOOSTING CLASSIFIER**

**DETEKSI HALAMAN WEBSITE PHISHING
MENGUNAKAN ALGORITMA MACHINE
LEARNING GRADIENT BOOSTING CLASSIFIER**

Penulis:

Roni Andarsyah,

Muhammad Rizal Supriadi

LOGO PENERBIT



DETEKSI HALAMAN WEBSITE PHISHING MENGGUNAKAN ALGORITMA MACHINE LEARNING GRADIENT BOOSTING CLASSIFIER

©Buku Pedia

Penulis:

Roni Andarsyah,
Muhammad Rizal Supriadi

Editor: Nisa Hanum Harani

Cetakan Pertama: Januari

Cover: Muhammad Rizal Supriadi Tata

Letak: Muhammad Rizal Supriadi

Hak Cipta 2023, pada Penulis. Diterbitkan pertama kali oleh:

Buku Pedia

Athena Residence Blok.E No. 1, Desa Ciwaruga, Kec. Parongpong, Kab.Bandung
Barat 40559

Website: bukupedia.co.id

E-mail: penerbit@bukupedia.co.id

Copyright © 2023 by Buku Pedia

All Right Reserved

- Cet. I —:Buku Pedia, 2023

ISBN

Hak cipta dilindungi undang-undang

Dilarang memperbanyak buku ini dalam bentuk dan dengan cara
apapun tanpa izin tertulis dari penulis dan penerbit

Undang-undang No.19 Tahun 2002 Tentang

Hak Cipta Pasal 72

Undang-undang No.19 Tahun 2002 Tentang Hak Cipta
Pasal 72

Barang siapa dengan sengaja melanggar dan tanpa hak melakukan perbuatan sebagaimana dimaksud dalam pasal ayat (1) atau pasal 49 ayat (1) dan ayat (2) dipidana dengan pidana penjara masing-masing paling sedikit 1 (satu) bulan dan/atau denda paling sedikit Rp.1.000.000,00 (satu juta rupiah), atau pidana penjara paling lama 7 (tujuh) tahun dan/atau denda paling banyak Rp.5.000.000.000,00 (lima miliar rupiah).

Barang siapa dengan sengaja menyiarkan, memamerkan, mengedarkan, atau menjual kepada umum suatu ciptaan atau barang hasil pelanggaran hak cipta terkait sebagai dimaksud pada ayat (1) dipidana dengan pidana penjara paling lama 5 (lima) tahun dan/atau denda paling banyak Rp.500.000.000,00 (lima ratus juta rupiah).

KATA PENGANTAR

Selamat datang di buku "Deteksi Halaman Website *Phishing* Menggunakan Algoritma Pengklasifikasi *Machine Learning Gradient Boosting Classifier*". Buku ini ditujukan bagi pemula yang ingin mempelajari cara membangun suatu sistem deteksi *website phishing* menggunakan teknik *machine learning*.

Buku ini akan memandu Anda melalui proses membangun sistem deteksi website phishing menggunakan algoritma pengklasifikasi machine learning *Gradient Boosting Classifier*. Anda akan belajar bagaimana menggunakan *python* untuk membuat model *machine learning* dan bagaimana menggunakan *framework flask* untuk membuat tampilan *website* yang interaktif dan *user friendly*. Setelah menyelesaikan tutorial ini, Anda akan memiliki keterampilan yang diperlukan untuk membangun sistem deteksi *website phishing* sendiri yang dapat membantu melindungi pengguna internet dari ancaman *phishing*.

Saya berharap buku ini dapat menjadi sumber informasi yang bermanfaat bagi pembaca!

Bandung, 18 Oktober 2021Penulis

Muhammad Rizal Supriadi

DAFTAR ISI

KATA PENGANTAR	i
DAFTAR ISI	ii
DAFTAR GAMBAR	v
BAB 1 Pendahuluan	1
1.1 Pengenalan Machine Learning.....	3
1.2 Supervised Learning	4
1.2.1 Logistic Regression.....	5
1.2.2 K-Nearest Neighbors.....	7
1.2.3 Support Vector Machine	9
1.2.4 Naïve Bayes	11
1.2.5 Decision Tree	13
1.2.6 Random Forest	15
1.2.7 Gradient Boosting	17
1.2.8 Catboost	19
1.3 Python.....	21
BAB 2 Pengenalan Tools	23
2.1 Framework Flask	23
2.2 Jupyter Notebook.....	24
2.3 Visual Studio Code	25
2.4 Pickle.....	26
2.5 Web Browser	27
BAB 3 Metodologi Penelitian	29
3.1 Diagram Alur Metodologi Penelitian	29

3.2 Tahapan – tahapan Diagram Alur	31
3.2.1 Sistem Evaluasi	32
3.2.2 Sistem Implementasi	32
3.3 Metodologi Data Science	31
BAB 4 Pembuatan Model Machine Learning.....	39
4.1 Import Library Tahap Awal	39
4.2 Memuat Dataset	42
4.3 Features Pada Dataset	42
4.3 Familiar dengan data & EDA	44
4.4 Memvisualisasikan Data	53
4.5 Memisahkan Dataset.....	56
4.6 Membangun dan Melatih Model	57
4.6.1 Model Logistic Regression.....	60
4.6.2 Model k-Nearest Neighbors.....	61
4.6.3 Model SVM.....	62
4.6.4 Model Naïve Bayes	64
4.6.5 Model Decision Tree	65
4.6.6 Model Random Forest	66
4.6.7 Model Gradient Boosting	67
4.6.8 Model Catboost	68
4.6.9 Model Multilayer Perceptrons	70
4.7 Pendandingan Model.....	71
4.8 Menyimpan Model Terbaik.....	73
4.9 Ubah Menjadi Format Pickle	74
4.10 Penelitian Sebelumnya	76
BAB 5 Pembuatan Aplikasi Web Phising	77
5.1 Install Flask.....	78

5.2	Membuat Struktur Folder	79
5.3	Import Pickle.....	79
5.4	Hasil.....	84
5.5	Kesimpulan	84
DAFTAR PUSTAKA.....		91

DAFTAR GAMBAR

<i>Gambar 1.1 Machine Learning</i>	3
<i>Gambar 1.2.1 Contoh Script Logistic Regression</i>	6
<i>Gambar 1.2.2 Contoh Script KNN</i>	8
<i>Gambar 1.2.3 Contoh Script SVM</i>	10
<i>Gambar 1.2.4 Contoh Script Naïve Bayes</i>	12
<i>Gambar 1.2.5 Contoh Script Decision Tree</i>	14
<i>Gambar 1.2.6 Contoh Script Random Forest</i>	16
<i>Gambar 1.2.7 Contoh Script Gradient Boosting</i>	18
<i>Gambar 1.2.8 Contoh Script Catboost</i>	20
<i>Gambar 2.1 Flask</i>	23
<i>Gambar 2.2 Jupyter Notebook</i>	24
<i>Gambar 2.3 VS Code</i>	25
<i>Gambar 3.1 Diagram Alur Metodologi Penelitian</i>	29
<i>Gambar 3.2.1 Flowchart Sistem Evaluasi</i>	32
<i>Gambar 3.2.2 Flowchart Sistem Impelemtasi</i>	35
<i>Gambar 4.1 Import Library Tahap Awal</i>	39
<i>Gambar 4.2.1 Memuat Datase</i>	44
<i>Gambar 4.2.2 Hasil Memuat Dataset</i>	44
<i>Gambar 4.3.1 Data Shape</i>	50
<i>Gambar 4.3.2 Data Columns</i>	50
<i>Gambar 4.3.3 Data Info</i>	51
<i>Gambar 4.3.4 Data Info</i>	52
<i>Gambar 4.4.1 Correlation heatmap</i>	53

<i>Gambar 4.4.2 Hasil Correlation heatmap.....</i>	<i>54</i>
<i>Gambar 4.4.3 Pairplot.....</i>	<i>54</i>
<i>Gambar 4.4.4 Hasil Pairplot.....</i>	<i>55</i>
<i>Gambar 4.4.4 Diagram Lingkaran.....</i>	<i>55</i>
<i>Gambar 4.4.5 Memisahkan Dataset.....</i>	<i>56</i>
<i>Gambar 4.6 Membuat Holder Model.....</i>	<i>59</i>
<i>Gambar 4.6.1 Model Logistic Regression.....</i>	<i>60</i>
<i>Gambar 4.6.2 Model KNN.....</i>	<i>61</i>
<i>Gambar 4.6.3 Model SVM.....</i>	<i>62</i>
<i>Gambar 4.6.4 Model Naïve Bayes.....</i>	<i>64</i>
<i>Gambar 4.6.5 Model Decision Tree.....</i>	<i>65</i>
<i>Gambar 4.6.6 Model Random Forest.....</i>	<i>66</i>
<i>Gambar 4.6.7 Model Gradient Boosting.....</i>	<i>67</i>
<i>Gambar 4.6.8 Model Catboost.....</i>	<i>68</i>
<i>Gambar 4.6.8 Model Multilayer Perceptrons.....</i>	<i>70</i>
<i>Gambar 4.7.1 Membuat Dataframe Perbandingan Model.....</i>	<i>71</i>
<i>Gambar 4.7.2 Sorting Dataframe Accuracy.....</i>	<i>72</i>
<i>Gambar 4.8 Menyimpan Model Terbaik.....</i>	<i>73</i>
<i>Gambar 4.9.1 Import Library pickle.....</i>	<i>74</i>
<i>Gambar 4.9.2 Cek Fitur pada Model.....</i>	<i>74</i>
<i>Gambar 4.9.3 Hasil Cek Fitur pada Model.....</i>	<i>75</i>
<i>Gambar 5.1 Script Flask.....</i>	<i>77</i>
<i>Gambar 5.1.1 Cek Flask Sudah Berjalan.....</i>	<i>78</i>
<i>Gambar 5.2.1 Struktur Folder Flask.....</i>	<i>79</i>
<i>Gambar 5.3.1 Import Library Flask.....</i>	<i>80</i>
<i>Gambar 5.3.2 Panggil Model.....</i>	<i>81</i>

Gambar 5.3.3 Ekstraksi Model 82

Gambar 5.3.4 Render Model 84

Gambar 5.4.1 Halaman Web Awal..... 85

Gambar 5.4.2 Check URL..... 86

Gambar 5.4.3 Hasil URL Phishing 88

Gambar 5.4.4 Hasil URL tidak Phishing 89

BAB 1

Pendahuluan

Dalam buku panduan/ tutorial ini, penulis akan memandu Anda melalui proses membangun suatu sistem Deteksi *Website Phising* yang menggunakan teknik *machine learning* dan *model Gradient Boosting Classifier*. Sistem ini akan dibuat menggunakan bahasa pemrograman *python*, yang akan digunakan untuk membuat model *machine learning* dan juga untuk membuat tampilan *website* menggunakan *framework flask*. Tujuan dari sistem ini adalah untuk membantu mengidentifikasi *website phising* yang mungkin merugikan pengguna internet dengan menipu mereka untuk memberikan informasi pribadi atau mengeluarkan dana dengan cara yang tidak sah.

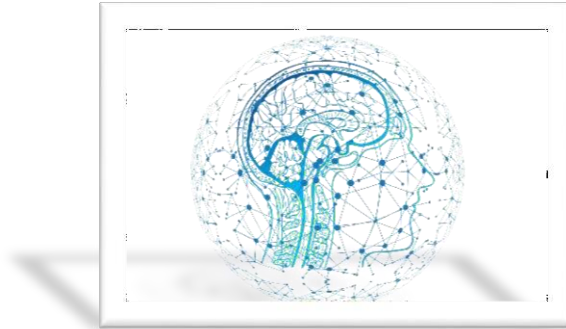
Phishing merupakan tindakan untuk mendapatkan informasi penting seseorang berupa *username*, *password* dan informasi sensitif lainnya dengan memberikan informasi palsu situs web yang mirip dengan aslinya. Pengelabuan (pemancing informasi penting) adalah sala satu bentuk tindak pidana yang bermaksud untuk mendapatkan rasahasia informasi dari seseorang, seperti nama pengguna, kata sandi dan kartu kredit, dengan menyamar sebagai orang lain atau bisnis terperaya di elektronik resmi komunikasi, seperti surat elektronik dan *instant messages*.^[1]

Dengan mengikuti tutorial ini, Anda akan belajar bagaimana menggunakan *python* untuk membuat model *machine learning*, serta bagaimana menggunakan *framework flask* untuk membuat tampilan *website* yang interaktif dan *user friendly*. Setelah menyelesaikan tutorial ini, Anda akan memiliki keterampilan yang diperlukan untuk membangun sistem Deteksi *Website Phishing* sendiri yang dapat membantu melindungi pengguna internet dari ancaman *phishing*.

Sebelum kita memulai pembangunan sistem deteksi *website phishing*, ada beberapa hal yang perlu dipertimbangkan. Pertama, pastikan bahwa Anda memiliki pengetahuan dasar tentang *machine learning* dan pengklasifikasi algoritma, karena kita akan menggunakan teknik ini untuk membangun sistem deteksi *website phishing*. Kedua, siapkan *tools* yang diperlukan untuk mendukung pembangunan sistem ini, seperti *python* untuk membuat model *machine learning* dan *framework flask* untuk membuat tampilan *website*.

Setelah Anda mempersiapkan hal-hal tersebut, kita dapat mulai membangun sistem deteksi *website phishing* sesuai dengan *tutorial* yang akan kita bahas di buku ini mulai dari perancangan, pembuatan model untuk menentukan model yang terbaik dalam melakukan deteksi *website phishing*, lalu melakukan *convert* model dan pembuatan aplikasi *website*.

1.1 Pengenalan Machine Learning



Gambar 1.1 Machine Learning

Pembelajaran mesin adalah aplikasi kecerdasan buatan (AI) yang memberikan sistem kemampuan untuk belajar dan belajar secara otomatis meningkatkan dari pengalaman tanpa diprogram secara eksplisit. Ini berfokus pada pengembangan program komputer yang bisa mengakses data dan menggunakannya untuk belajar sendiri.

Algoritma pembelajaran mesin sering dikategorikan sebagai diawasi atau tidak diawasi. Algoritma yang diawasi membutuhkan ilmuwan data atau analis data dengan keterampilan pembelajaran mesin untuk memberikan *input* dan yang diinginkan *output*, selain memberikan umpan balik tentang keakuratan prediksi selama pelatihan algoritma [2].

Berikut ini adalah beberapa contoh penerapan *machine learning*:

1. Sistem rekomendasi: Sistem rekomendasi

menggunakan *machine learning* untuk mengelompokkan pengguna berdasarkan preferensi dan kebiasaan mereka, dan kemudian menyarankan produk atau layanan yang mungkin mereka sukai.

2. Analisis sentimen: *Machine learning* dapat digunakan untuk menganalisis sentimen terhadap suatu produk atau layanan dengan mempelajari ulasan atau komentar yang diberikan oleh pengguna.
3. Pendeteksi spam: *Machine learning* dapat digunakan untuk mengidentifikasi dan memblokir *email* spam dengan mempelajari pola-pola yang biasanya terdapat pada *email* spam.
4. Pengenalan wajah: *Machine learning* dapat digunakan untuk mengenali wajah orang dalam foto atau video dengan mempelajari wajah-wajah yang telah dikenali terlebih dahulu.
5. Penerjemahan otomatis: *Machine learning* dapat digunakan untuk menerjemahkan teks dari satu bahasa ke bahasa lain dengan mempelajari terjemahan yang telah dibuat oleh manusia.

1.2 Supervised Learning

Algoritma pembelajaran mesin *Supervised Learning* dapat menerapkan apa yang telah dipelajari di masa lalu ke data baru menggunakan contoh berlabel memprediksi peristiwa masa depan. Dimulai dari

analisis dataset pelatihan yang diketahui, algoritma pembelajaran menghasilkan kesimpulan berfungsi untuk membuat prediksi tentang nilai output. Sistem mampu memberikan target untuk setiap masukan baru setelah cukup pelatihan. Algoritma pembelajaran juga dapat membandingkan keluarannya dengan keluaran yang benar dan diinginkan serta menemukan kesalahan untuk dimodifikasi sesuai dengan modelnya.[3]

1.2.1 Logistic Regression

Regresi logistik adalah model prediktif yang digunakan untuk mengevaluasi hubungan antara variabel dependen (target) yang merupakan data kategorikal dengan skala nominal atau ordinal dan variabel independen (prediktor) yang merupakan data kategorikal dengan skala interval atau rasio. Algoritma ini juga dapat digunakan untuk pemodelan deret waktu untuk menemukan hubungan antar variabel yang terlibat. Regresi logistik adalah algoritma yang digunakan untuk memprediksi probabilitas variabel dependen kategori. Dalam regresi logistik, variabel dependen ditampilkan sebagai variabel biner yang bernilai 1 (ya) atau 0 (tidak). Model regresi logistik memprediksi sebagai fungsi X . Asumsi yang digunakan dalam regresi Logistik adalah sebagai

berikut: regresi logistik biner membutuhkan variabel dependen biner, untuk regresi biner, tingkat faktor 1 dari variabel dependen harus mewakili hasil yang diinginkan, variabel independen harus independen satu sama lain. Dalam hal ini, model harus memiliki sedikit atau tidak ada multikolinearitas dan berhubungan secara linear dengan peluang log [4].

Berikut ini adalah contoh script sederhana untuk melakukan *logistic regression* menggunakan *Python*:

```
# Import library yang dibutuhkan
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

# Persiapkan data
X = [[2, 3], [4, 5], [6, 7], [8, 9]] # data fitur
y = [0, 0, 1, 1] # data kelas

# Split data menjadi data training dan data testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

# Buat model Logistic regression
model = LogisticRegression()

# Latih model dengan data training
model.fit(X_train, y_train)

# Uji model dengan data testing
score = model.score(X_test, y_test)

# Cetak akurasi model
print("Akurasi model:", score)

# Prediksi kelas suatu objek baru
prediction = model.predict([[5, 6]])
print("Kelas prediksi:", prediction)
```

Gambar 1.2.1 Contoh Script Logistic Regression

Penjelasan *script Logistic Regression* diatas: *Script* tersebut terdiri dari beberapa langkah:

1. *Import library* yang dibutuhkan, yaitu

LogisticRegression dan *train_test_split* dari *scikit-learn*.

2. Persiapkan data fitur dan kelas yang akan digunakan untuk melatih model.
3. *Split* data menjadi data training dan data *testing* dengan ukuran data *testing* sebesar 30%.
4. Buat model *logistic regression*.
5. Latih model dengan data *training*.
6. Uji model dengan data *testing* dan dapatkan skor akurasi.
7. Cetak akurasi model.
8. Prediksi kelas suatu objek baru.
9. Cetak kelas prediksi.

Setelah melakukan semua langkah tersebut, Anda akan mendapatkan model *logistic regression* yang telah dilatih dan diuji serta dapat digunakan untuk melakukan prediksi kelas suatu objek baru.

1.2.2 K-Nearest Neighbors

K-Nearest Neighbor adalah metode klasifikasi dengan mencari jarak terdekat antara data yang akan dievaluasi dengan *K-Nearest Neighbors* terdekatnya dalam data pelatihan. Model ini dapat digunakan dalam klasifikasi yang akan dilakukan data *training* didalam proses pelatihan tersebut.[5] Proses pelatihan KNN

menghasilkan k yang memberikan akurasi tertinggi dalam menggeneralisasi data yang akan datang.

Berikut ini adalah contoh *script* sederhana untuk melakukan *K-Nearest Neighbors* (KNN) menggunakan *Python*:

```
# Import Library yang dibutuhkan
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

# Persiapkan data
X = [[2, 3], [4, 5], [6, 7], [8, 9]] # data fitur
y = [0, 0, 1, 1] # data kelas

# Split data menjadi data training dan data testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

# Buat model KNN dengan jumlah tetangga sebanyak 3
model = KNeighborsClassifier(n_neighbors=3)

# Latih model dengan data training
model.fit(X_train, y_train)

# Uji model dengan data testing
score = model.score(X_test, y_test)

# Cetak akurasi model
print("Akurasi model:", score)

# Prediksi kelas suatu objek baru
prediction = model.predict([[5, 6]])
print("Kelas prediksi:", prediction)
```

Gambar 1.2.2 Contoh Script KNN

Penjelasan *script* KNN diatas:

Script tersebut terdiri dari beberapa langkah:

1. *Import library* yang dibutuhkan, yaitu ***KNeighborsClassifier*** dan ***train_test_split*** dari *scikit-learn*.
2. Persiapkan data fitur dan kelas yang akan digunakan untuk melatih model.

3. *Split* data menjadi data training dan data *testing* dengan ukuran data *testing* sebesar 30%.
4. Buat model KNN.
5. Latih model dengan data *training*.
6. Uji model dengan data *testing* dan dapatkan skor akurasi.
7. Cetak akurasi model.
8. Prediksi kelas suatu objek baru.
9. Cetak kelas prediksi.

Setelah melakukan semua langkah tersebut, Anda akan mendapatkan model KNN yang telah dilatih dan diuji serta dapat digunakan untuk melakukan prediksi kelas suatu objek baru.

1.2.3 Support Vector Machine

Mesin vektor pendukung adalah algoritma kuat lainnya di teknologi pembelajaran mesin. Dalam mendukung mesin vektor algoritma setiap item data diplot sebagai titik dalam n-dimensi ruang dan mendukung konstruksi algoritma mesin vector garis pemisah untuk klasifikasi dua kelas, pemisahan ini garis dikenal sebagai *hyperplane*.

SVM adalah teknik pembelajaran mesin berdasarkan *Supervised Learning* dan sesuai untuk kedua regresi dan klasifikasi. SVM dianggap sebagai pencapaian teknik modern penerimaan cepat karena

hasil yang baik dicapai dalam banyak bidang masalah data mining, berdasarkan fondasi yang kuat dalam teori belajar statistik. SVM adalah teknik klasifikasi berdasarkan pembelajaran statistik, yang berhasil dimanfaatkan dalam banyak aplikasi klasifikasi nonlinier dan besar dataset dan masalah. [6].

Berikut ini adalah contoh script sederhana untuk melakukan *Support Vector Machine* (SVM) menggunakan *Python*:

```
# Import library yang dibutuhkan
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split

# Persiapkan data
X = [[2, 3], [4, 5], [6, 7], [8, 9]] # data fitur
y = [0, 0, 1, 1] # data kelas

# Split data menjadi data training dan data testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

# Buat model SVM dengan kernel linear
model = SVC(kernel='linear')

# Latih model dengan data training
model.fit(X_train, y_train)

# Uji model dengan data testing
score = model.score(X_test, y_test)

# Cetak akurasi model
print("Akurasi model:", score)

# Prediksi kelas suatu objek baru
prediction = model.predict([[5, 6]])
print("Kelas prediksi:", prediction)
```

Gambar 1.2.3 Contoh Script SVM

Penjelasan *script* SVM diatas:

Script tersebut terdiri dari beberapa langkah:

1. *Import library* yang dibutuhkan, yaitu **SVC** dan

train_test_split dari *scikit-learn*.

2. Persiapkan data fitur dan kelas yang akan digunakan untuk melatih model.
3. *Split* data menjadi data training dan data *testing* dengan ukuran data *testing* sebesar 30%.
4. Buat model SVM.
5. Latih model dengan data *training*.
6. Uji model dengan data *testing* dan dapatkan skor akurasi.
7. Cetak akurasi model.
8. Prediksi kelas suatu objek baru.
9. Cetak kelas prediksi.

Setelah melakukan semua langkah tersebut, Anda akan mendapatkan model SVM yang telah dilatih dan diuji serta dapat digunakan untuk melakukan prediksi kelas suatu objek baru.

1.2.4 Naïve Bayes

Naive bayes termasuk ke dalam pembelajaran *supervised*, sehingga pada tahapan pembelajaran dibutuhkan data awal berupa data pelatihan untuk dapat mengambil keputusan. Pada tahapan pengklasifikasian akan dihitung nilai probabilitas dari masing-masing label kelas yang ada terhadap masukan yang diberikan.

Pengklasifikasi *Naïve Bayes* adalah salah satu

deteksi tinggi pendekatan untuk mempelajari klasifikasi dokumen teks. Diberikan satu set sampel pelatihan rahasia, sebuah aplikasi dapat belajar dari sampel tersebut, sehingga dapat memprediksi kelas sampel yang tidak terpenuhi. [7].

Berikut ini adalah contoh *script* sederhana untuk melakukan *Naive Bayes* menggunakan *Python*:

```
# Import library yang dibutuhkan
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split

# Persiapkan data
X = [[2, 3], [4, 5], [6, 7], [8, 9]] # data fitur
y = [0, 0, 1, 1] # data kelas

# Split data menjadi data training dan data testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

# Buat model Naive Bayes
model = GaussianNB()

# Latih model dengan data training
model.fit(X_train, y_train)

# Uji model dengan data testing
score = model.score(X_test, y_test)

# Cetak akurasi model
print("Akurasi model:", score)

# Prediksi kelas suatu objek baru
prediction = model.predict([[5, 6]])
print("Kelas prediksi:", prediction)
```

Gambar 1.2.4 Contoh Script *Naive Bayes*

Penjelasan *script Naive Bayes* diatas:

Script tersebut terdiri dari beberapa langkah:

1. *Import library* yang dibutuhkan, yaitu ***GaussianNB*** dan ***train_test_split*** dari *scikit-learn*.
2. Persiapkan data fitur dan kelas yang akan digunakan untuk melatih model.

3. *Split* data menjadi data training dan data *testing* dengan ukuran data *testing* sebesar 30%.
4. Buat model *Naïve Bayes*.
5. Latih model dengan data *training*.
6. Uji model dengan data *testing* dan dapatkan skor akurasi.
7. Cetak akurasi model.
8. Prediksi kelas suatu objek baru.
9. Cetak kelas prediksi.

Setelah melakukan semua langkah tersebut, Anda akan mendapatkan model *Naïve Bayes* yang telah dilatih dan diuji serta dapat digunakan untuk melakukan prediksi kelas suatu objek baru.

1.2.5 Decision Tree

Pohon keputusan (DT). DT mengklasifikasikan barang berdasarkan pembuatan keputusan pada setiap cabang untuk mendapatkan sebanyak- banyaknya keuntungan entropi sebanyak mungkin. Sebuah pohon keputusan terdiri dari a simpul akar, beberapa simpul internal, dan simpul daun. Daun *node* menunjukkan hasil dari *classifier*, dan lainnya *node* menunjukkan setiap atribut. Setiap rute dari simpul akar ke simpul daun sesuai dengan penentuan urutan pengujian. Ini mengikuti aturan dari memecah dan menaklukkan [8]. Setiap pohon memiliki

cabang, cabang mewakili suatu atribut yang harus dipenuhi untuk menuju cabang selanjutnya hingga berakhir di daun (tidak ada cabang lagi). Konsep data dalam *decision tree* adalah data dinyatakan dalam bentuk tabel yang terdiri dari atribut dan *record*. Atribut digunakan sebagai parameter yang dibuat sebagai kriteria dalam pembuatan pohon.

Berikut ini adalah contoh script sederhana untuk melakukan *decision tree* menggunakan *Python*:

```
# Import library yang dibutuhkan
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split

# Persiapkan data
X = [[2, 3], [4, 5], [6, 7], [8, 9]] # data fitur
y = [0, 0, 1, 1] # data kelas

# Split data menjadi data training dan data testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

# Buat model decision tree dengan kriteria gini
model = DecisionTreeClassifier(criterion='gini')

# Latih model dengan data training
model.fit(X_train, y_train)

# Uji model dengan data testing
score = model.score(X_test, y_test)

# Cetak akurasi model
print("Akurasi model:", score)

# Prediksi kelas suatu objek baru
prediction = model.predict([[5, 6]])
print("Kelas prediksi:", prediction)
```

Gambar 1.2.5 Contoh Script Decision Tree

Penjelasan *script Decision Tree* diatas:

Script tersebut terdiri dari beberapa langkah:

1. *Import library* yang dibutuhkan, yaitu

DecisionTreeClassifier dan *train_test_split* dari *scikit-learn*.

2. Persiapkan data fitur dan kelas yang akan digunakan untuk melatih model.
3. *Split* data menjadi data training dan data *testing* dengan ukuran data *testing* sebesar 30%.
4. Buat model *Decision Tree*.
5. Latih model dengan data *training*.
6. Uji model dengan data *testing* dan dapatkan skor akurasi.
7. Cetak akurasi model.
8. Prediksi kelas suatu objek baru.
9. Cetak kelas prediksi.

Setelah melakukan semua langkah tersebut, Anda akan mendapatkan model *Decision Tree* yang telah dilatih dan diuji serta dapat digunakan untuk melakukan prediksi kelas suatu objek baru.

1.2.6 Random Forest

Algoritma *Random Forest* dapat mencapai akurasi tertinggi sebelum dan sesudah pemilihan fitur dan peningkatan bangunan secara dramatis. Hasil percobaan menunjukkan bahwa menggunakan pendekatan seleksi dengan mesin algoritma pembelajaran dapat meningkatkan efektivitas model klasifikasi untuk deteksi *phishing* tanpa mengurangi

kinerja mereka. [9]. Dalam *random forest*, banyak pohon ditanam sehingga terbentuk hutan (*forest*), kemudian analisis dilakukan pada kumpulan pohon tersebut.

Berikut ini adalah contoh *script* sederhana untuk melakukan random forest menggunakan *Python*:

```
# Import library yang dibutuhkan
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

# Persiapkan data
X = [[2, 3], [4, 5], [6, 7], [8, 9]] # data fitur
y = [0, 0, 1, 1] # data kelas

# Split data menjadi data training dan data testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

# Buat model random forest dengan jumlah pohon sebanyak 10
model = RandomForestClassifier(n_estimators=10)

# Latih model dengan data training
model.fit(X_train, y_train)

# Uji model dengan data testing
score = model.score(X_test, y_test)

# Cetak akurasi model
print("Akurasi model:", score)

# Prediksi kelas suatu objek baru
prediction = model.predict([[5, 6]])
print("Kelas prediksi:", prediction)
```

Gambar 1.2.6 Contoh Script Random Forest

Penjelasan *script* Random Forest diatas:

Script tersebut terdiri dari beberapa langkah:

1. *Import library* yang dibutuhkan, yaitu ***RandomForestClassifier*** dan ***train_test_split*** dari *scikit-learn*.
2. Persiapkan data fitur dan kelas yang akan

digunakan untuk melatih model.

3. *Split* data menjadi data training dan data *testing* dengan ukuran data *testing* sebesar 30%.
4. Buat model *Random Forest*.
5. Latih model dengan data *training*.
6. Uji model dengan data *testing* dan dapatkan skor akurasi.
7. Cetak akurasi model.
8. Prediksi kelas suatu objek baru.
9. Cetak kelas prediksi.

Setelah melakukan semua langkah tersebut, Anda akan mendapatkan model *Random Forest* yang telah dilatih dan diuji serta dapat digunakan untuk melakukan prediksi kelas suatu objek baru.

1.2.7 Gradient Boosting

Tujuan utama dari *Boosting* adalah menggabungkan semua train yang lemah bersama-sama untuk membentuk model yang kuat.

- *Gradient boosting* adalah suatu teknik yang sangat kuat untuk mengembangkan model prediktif. Ini berlaku untuk beberapa fungsi risiko dan mengoptimalkan akurasi prediksi model. Ini juga menyelesaikan masalah multikolinearitas di mana korelasi antar variabel prediktor tinggi.

- *Gradient Boosting* adalah algoritma pembelajaran mesin ansambel dan biasanya digunakan untuk menyelesaikan klasifikasi dan regresi [10].

Berikut ini adalah contoh *script* sederhana untuk melakukan *gradient boosting* menggunakan *Python*:

```
# Import library yang dibutuhkan
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import train_test_split

# Persiapkan data
X = [[2, 3], [4, 5], [6, 7], [8, 9]] # data fitur
y = [0, 0, 1, 1] # data kelas

# Split data menjadi data training dan data testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

# Buat model gradient boosting dengan jumlah pohon sebanyak 10
model = GradientBoostingClassifier(n_estimators=10)

# Latih model dengan data training
model.fit(X_train, y_train)

# Uji model dengan data testing
score = model.score(X_test, y_test)

# Cetak akurasi model
print("Akurasi model:", score)

# Prediksi kelas suatu objek baru
prediction = model.predict([[5, 6]])
print("Kelas prediksi:", prediction)
```

Gambar 1.2.7 Contoh Script Gradient Boosting

Penjelasan *script Gradient Boosting* diatas: *Script*

tersebut terdiri dari beberapa langkah:

1. *Import library* yang dibutuhkan, yaitu ***GradientBoostingClassifier*** dan ***train_test_split*** dari *scikit-learn*.
2. Persiapkan data fitur dan kelas yang akan digunakan untuk melatih model.
3. *Split* data menjadi data training dan data *testing*

dengan ukuran data *testing* sebesar 30%.

4. Buat model *Gradient Boosting*.
5. Latih model dengan data *training*.
6. Uji model dengan data *testing* dan dapatkan skor akurasi.
7. Cetak akurasi model.
8. Prediksi kelas suatu objek baru.
9. Cetak kelas prediksi.

Setelah melakukan semua langkah tersebut, Anda akan mendapatkan model *Gradient Boosting* yang telah dilatih dan diuji serta dapat digunakan untuk melakukan prediksi kelas suatu objek baru.

1.2.8 Catboost

Algoritma *CatBoost* membutuhkan lebih banyak waktu untuk pelatihan dan menguji kumpulan data yang menggunakan lebih banyak komputasi sumber daya. Efektivitas algoritma *CatBoost* memiliki telah ditunjukkan melalui kinerjanya yang lebih tinggi algoritma yang bersaing. Dalam hal pekerjaan masa depan, *Apache* Kerangka kerja Spark dapat digunakan untuk meningkatkan *Sickit-learn library* yang disebut *Sk-dist*. *Sk-dist* telah mengatasi batasantersebut perpustakaan *Sickit-learn* seperti memakan waktu dan lagging pelatihan model [11].

Peneliti dapat mengatur pengaturan untuk jumlah

maksimum iterasi yang digunakan *CatBoost*, kedalaman maksimum Pohon Keputusan konstituen, dan jumlah maksimum kombinasi fitur kategorikal untuk meningkatkan performa model. Nilai-nilai itu peneliti menggunakan *hyper-parameter* ini dapat menjelaskan perbedaan dalam kinerja *CatBoost*.

Berikut ini adalah contoh script sederhana untuk melakukan *CatBoost* menggunakan *Python*:

```
# Import library yang dibutuhkan
from catboost import CatBoostClassifier
from sklearn.model_selection import train_test_split

# Persiapkan data
X = [[2, 3], [4, 5], [6, 7], [8, 9]] # data fitur
y = [0, 0, 1, 1] # data kelas

# Split data menjadi data training dan data testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

# Buat model CatBoost
model = CatBoostClassifier()

# Latih model dengan data training
model.fit(X_train, y_train)

# Uji model dengan data testing
score = model.score(X_test, y_test)

# Cetak akurasi model
print("Akurasi model:", score)

# Prediksi kelas suatu objek baru
prediction = model.predict([[5, 6]])
print("Kelas prediksi:", prediction)
```

Gambar 1.2.8 Contoh Script Catboost

Penjelasan *script Catboost* diatas:

Script tersebut terdiri dari beberapa langkah:

1. *Import library* yang dibutuhkan, yaitu *CatBoostClassifier* dan *train_test_split* dari *scikit-*

learn.

2. Persiapkan data fitur dan kelas yang akan digunakan untuk melatih model.
3. *Split* data menjadi data training dan data *testing* dengan ukuran data *testing* sebesar 30%.
4. Buat model *Cat Boost*.
5. Latih model dengan data *training*.
6. Uji model dengan data *testing* dan dapatkan skor akurasi.
7. Cetak akurasi model.
8. Prediksi kelas suatu objek baru.
9. Cetak kelas prediksi.

Setelah melakukan semua langkah tersebut, Anda akan mendapatkan model *Cat Boost* yang telah dilatih dan diuji serta dapat digunakan untuk melakukan prediksi kelas suatu objek baru.

1.3 Python

Python adalah bahasa yang dirancang dengan baik yang dapat digunakan secara nyata pemrograman dunia. *Python* adalah tingkat yang sangat tinggi, dinamis, berorientasi objek, bahasa pemrograman tujuan umum yang menggunakan juru bahasa dan dapat digunakan dalam domain yang luas aplikasi. *Python* dirancang agar mudah dimengerti dan gunakan. *Python* disebut sebagai sangat *user-friendly* dan bahasa yang

ramah pemula belakangan ini. *Python* punya memperoleh popularitas karena menjadi bahasa yang ramah bagi pemula, dan telah menggantikan Java sebagai pengantar paling populer bahasa [12].

BAB 2

Pengenalan Tools

Setelah Anda memahami materi pengenalan dasar tentang *machine learning* dan pengklasifikasi algoritma yang akan kita gunakan dalam membangun sistem deteksi *website phishing*, selanjutnya Anda perlu mengetahui *tools* apa saja yang akan digunakan dalam proses pembangunan sistem ini. Pastikan bahwa Anda sudah memasang *tools* tersebut di komputer Anda sebelum memulai pembangunan sistem deteksi *website phishing* sesuai dengan tutorial yang akan kita bahas di buku ini.

2.1 Framework Flask



Gambar 2.1 Flask

Flask adalah sebuah web *framework* yang ditulis dengan bahasa Python dan tergolong sebagai jenis *microframework*. Flask berfungsi sebagai kerangka kerja aplikasi dan tampilan dari suatu web. Dengan menggunakan *Flask* dan bahasa *Python*, pengembang

dapat membuat sebuah web yang terstruktur dan dapat mengatur behaviour suatu web dengan lebih mudah. *Flask* termasuk pada jenis *microframework* karena tidak memerlukan suatu alat ataupun pustaka tertentu dalam penggunaannya. Sebagian besar fungsi dan komponen umum seperti validasi form, *database*, dan sebagainya tidak terpasang secara default di *Flask* [13]. Hal ini dikarenakan fungsi dan komponen-komponen tersebut sudah disediakan oleh pihak ketiga dan *Flask* dapat menggunakan ekstensi yang membuat fitur dan komponen-komponen tersebut seakan diimplementasikan oleh Flask sendiri.

2.2 Jupyter Notebook



Gambar 2.2 Jupyter Notebook

Jupyter Notebook (file yang berekstensi ipynb) adalah dokumen yang dihasilkan oleh *Jupyter Notebook App* yang berisikan kode komputer dan rich text element seperti paragraf, persamaan matematik, gambar dan tautan (links). [14]

Jupyter Notebook tersedia dalam berbagai bahasa pemrograman, termasuk *python*, R, Julia, dan banyak lagi. Anda dapat menggunakan *Jupyter Notebook* untuk menulis dan menjalankan kode *python* secara interaktif, serta memvisualisasikan data dan hasilnya dengan mudah.

Untuk menggunakan *Jupyter Notebook*, Anda perlu memasang aplikasinya terlebih dahulu. Anda dapat mengikuti instruksi pemasangan di halaman resmi *Jupyter Notebook* di <https://jupyter.org/install>. Setelah terpasang, Anda dapat membuka *Jupyter Notebook* dari command prompt atau terminal dengan mengetik perintah "*jupyter notebook*" dan mengikuti instruksi selanjutnya untuk membuat dan menjalankan *notebook*.

2.3 Visual Studio Code



Gambar 2.3 VS Code

Visual Studio Code (VS Code) ini adalah sebuah teks editor ringan dan handal yang dibuat oleh *Microsoft* untuk sistem operasi *multiplatform*, artinya tersedia juga untuk versi *Linux*, *Mac*, dan *Windows*. Teks editor ini secara langsung mendukung bahasa pemrograman *JavaScript*, *Typescript*, dan *Node.js*, serta bahasa pemrograman lainnya dengan bantuan plugin yang dapat dipasang via *marketplace Visual Studio Code* (seperti *C++*, *C#*, *Python*, *Go*, *Java*, dst).

Banyak sekali fitur-fitur yang disediakan oleh *Visual Studio Code*, diantaranya *Intellisense*, *Git Integration*, *Debugging*, dan fitur ekstensi yang menambah kemampuan teks editor. Fitur-fitur tersebut akan terus bertambah seiring dengan bertambahnya versi *Visual Studio Code*. Pembaruan versi *Visual Studio Code* ini juga dilakukan berkala setiap bulan, dan inilah yang membedakan *VS Code* dengan teks editor-teks editor yang lain. [15]

2.4 Pickle

Modul *pickle* mengimplementasikan protokol biner untuk serialisasi dan *de-serialisasi* struktur objek *Python*. "*Pickling*" adalah proses di mana hierarki objek *Python* diubah menjadi aliran *byte*, dan "*unpickling*"

adalah operasi kebalikannya, di mana aliran *byte* (dari file biner atau objek mirip *byte*) diubah kembali menjadi hierarki objek.

Library python pickle, yang mengonversi hierarki objek Python ke dan dari aliran byte khusus *Python* (proses masing-masing dikenal sebagai 'pengawetan' dan 'pembongkaran'). Ada beberapa protokol yang berbeda, dan file tidak dirancang agar kompatibel antara versi *Python*, atau dapat ditafsirkan dengan versi lain bahasa [16]

2.5 Web Browser

Web browser disebut juga sebagai perambah, adalah perangkat lunak yang berfungsi menampilkan dan melakukan interaksi dengan dokumen-dokumen yang disediakan oleh *server web*. *Browser* pada umumnya juga mendukung berbagai jenis URL dan protokol, misalnya ftp: untuk *file transfer protocol* (FTP), rtsp: untuk *real-time streaming protocol* (RTSP), and https: untuk versi http yang terenkripsi (SSL). File format sebuah halaman web biasanya *hyper-text markup language* (HTML) dan diidentifikasi dalam protokol HTTP menggunakan *header MIME*, format lainnya antara lain XML dan XHTML. Sebagian besar browser mendukung bermacam format tambahan pada HTML seperti format. gambar JPEG, PNG and GIF

image formats, dan dapat dikembangkan dukungannya misal terhadap SVG dengan menambahkan/ menggunakan *plugin*. Ada beberapa web browser yang populer diantaranya *Internet Explorer*, Crome, Opera dan Mozilla. [17]

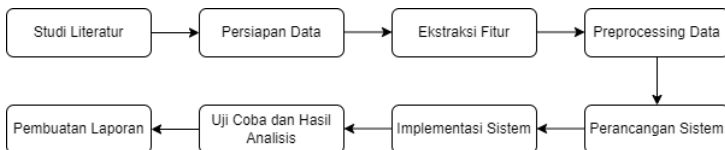
BAB 3

Metodologi Penelitian

3.1 Diagram Alur Metodologi Penelitian

Sistem yang diusulkan pada penelitian ini terdiri dari dua sistem utama, yaitu Sistem Evaluasi dan Implementasi Sistem, dimana implementasi sistem mengandung proses penting, yaitu fitur ekstraksi, sementara pada Sistem Evaluasi akan ada menjadi proses pengoptimalan parameter dari algoritma klasifikasi yang akan digunakan. Algoritma yang akan digunakan dalam penelitian ini adalah *Logistic Regression*, *K-Nearest Neighbors*, *Support Vector Machine*, *Naïve Bayes*, *Decision Tree*, *Random Forest*, *Gradient Boosting* dan *Catboost* sebagai perbandingan kinerja sistem.

Tahapan yang akan dilakukan dalam penelitian ini dapat dilihat pada gambar berikut:



Gambar 3.1 Diagram Alur Metodologi Penelitian

Berdasarkan diagram pada diatas, secara umum penelitian dapat digambarkan sebagai berikut:

1. Studi Literatur merupakan tahapan awal dari penelitian ini. Tahapan ini dilakukan untuk mengumpulkan penelitian yang berkaitan dengan metode yang digunakan kepada ekstraksi fitur dan klasifikasi.
2. Persiapan data merupakan langkah yang dilakukan untuk mendapatkan dataset yang akan digunakan dalam mengklasifikasikan *website*. Dataset yang digunakan bersumber dari *Kaggle*.
3. Ekstraksi fitur dilakukan untuk mengekstraksi fitur terdapat di situs web berdasarkan dataset yang diperoleh dari *Kaggle*. Hasil ekstraksi fitur ini kemudian akan digunakan untuk mendeteksi situs web *phishing*.
4. *Preprocessing* Data dilakukan dalam bentuk menganalisis data yang akan digunakan untuk memilih data berdasarkan hasil ekstraksi ciri yang telah dilakukan sebelumnya.
5. Perancangan sistem pada penelitian ini dilakukan untuk menentukan algoritma yang akan digunakan dalam klasifikasi situs web *phishing*. Algoritma- algoritma yang digunakan dalam mengklasifikasi adalah *Logistic Regression*, *K-Nearest Neighbors*, *Support Vector Machine*, *Naïve Bayes*, *Decision Tree*, *Random Forest*, *Gradient Boosting* dan *Catboost* sebagai perbandingan kinerja sistem.

Pada tahapan ini dibuat *flowchart* yang berkaitan dengan alur kerja sistem.

6. Implementasi sistem dilakukan sesuai dengan *flowchart* yang telah dibuat ditahap sebelumnya. Pada penelitian ini, sistem dibuat dengan menggunakan bahasa pemrograman *python* dan *framework flask*.
7. Pengujian sistem dilakukan untuk mengetahui keakuratan sistem yang dibuat, sistem akan diuji dengan beberapa percobaan dan bentuk URL yang berbeda.
8. Tahapan akhir dalam penelitian ini adalah menulis laporan penelitian dalam bentuk laporan.

3.2 Tahapan – tahapan Diagram Alur Metofologi Penelitian Pada

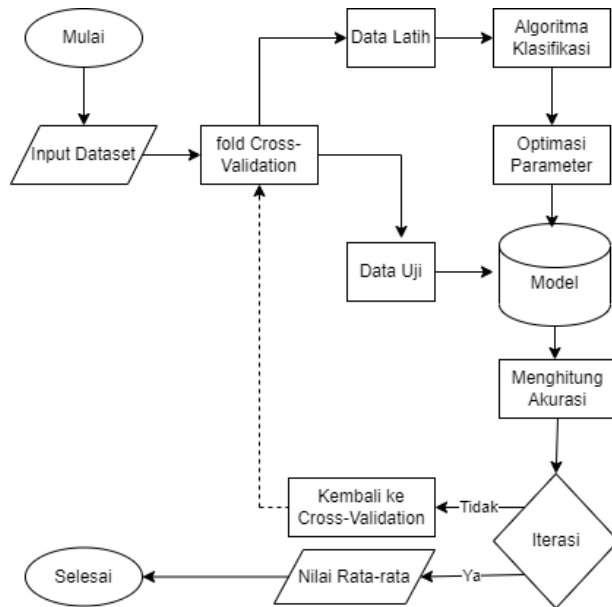
tahapan penelitian ini dilakukan untuk mengekstrak fitur yang terdapat pada situs web dengan memasukkan URL ke dalam sistem dankemudian sistem akan mengkase URL dan kemudian istem akan mengakses URL dan melakukan fitur ekstraksi di situs *website*. Karena untuk mengakses situs web yang akandilakukan deteksi harus terhubung kedalam internet. Sistem ini secara otomatis menjadi sistem online.

Sistem yang dibuat terdiri dari dua bagian yaitu sistem pelaksanaan dan sistem evaluasi. Implementasi sistem dapat digunakan ketika pengguna penginputkan

URL kedalam form. Sedangkan sistem evaluasi adalah sistem dibuat untuk menganalisis kinerja dari sistem implementasi.

3.2.1 Sistem Evaluasi

Sistem evaluasi merupakan sistem yang dirancang sebagai untuk evaluasi sistem produksi. Evaluasi terhadap sistem ini dilakukan dengan cara mengevaluasi dataset yang digunakan dalam implementasi sistem. *Flowchart* sistem evaluasi dapat dilihat pada gambar berikut:



Gambar 3.2.1 Flowchart Sistem Evaluasi

Pada tahapan diatas, proses pembuatan model deteksi phishing dimulai dari input dataset.

Dataset yang digunakan berisi informasi tentang URL situs web beserta label kelas yang mengidentifikasinya sebagai situs web *phishing* atau bukan. Setelah dataset diinput, proses selanjutnya adalah melakukan *fold Cross Validation*.

Cross Validation adalah metode untuk mengevaluasi kinerja model dengan membagi dataset menjadi beberapa bagian yang disebut sebagai fold. Dari beberapa fold tersebut, satu fold digunakan sebagai data pengujian dan sisanya digunakan sebagai data latih. Proses ini dilakukan beberapa kali dengan menggunakan setiap fold sebagai data pengujian. Hal ini dilakukan untuk mengevaluasi kinerja model dengan menghitung rata-rata akurasi dari setiap fold.

Setelah proses *fold Cross Validation* selesai, data tersebut dilatih menggunakan algoritma klasifikasi. Algoritma yang digunakan dapat berbeda-beda tergantung pada kebutuhan dan karakteristik dari dataset yang digunakan. Selanjutnya, dilakukan optimasi parameter untuk meningkatkan kinerja model.

Pengujian untuk pembuatan model. Pengujian ini dilakukan dengan membandingkan kinerja model yang telah dilatih dengan dataset pengujian yang tidak digunakan sebelumnya. Hal ini dilakukan untuk

mengevaluasi kinerja model dengan data yang belum pernah dilihat sebelumnya.

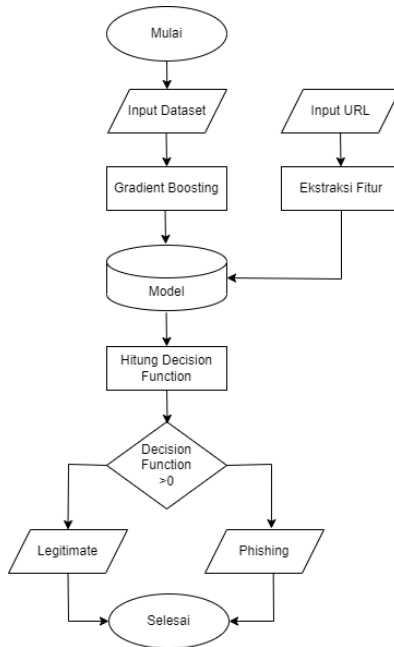
Setelah proses pengujian selesai, dilakukan penghitungan akurasi dari model yang telah dibuat. Akurasi adalah seberapa baik model dapat mengklasifikasikan data. Semakin tinggi nilai akurasi, semakin baik kinerja model. Namun, dalam beberapa kasus, akurasi tidak cukup untuk menentukan kinerja model yang baik.

Selanjutnya, dilakukan tahapan iterasi. Tahapan ini dilakukan untuk meningkatkan kinerja model dengan cara mengubah beberapa parameter atau mencoba algoritma klasifikasi yang berbeda. Proses iterasi ini dilakukan sampai kita dapat menentukan nilai rata-rata akurasi yang diinginkan. Jika nilai rata-rata akurasi sudah dapat ditentukan, maka program akan selesai dan model yang telah dibuat dapat digunakan untuk melakukan deteksi *phishing*.

3.2.2 Sistem Implementasi

Sistem implementasi merupakan sistem yang dirancang untuk digunakan dan diimplementasikan dalam melakukan deteksi website *phishing*. Hal ini yang membedakan sistem implementasi dengan sistem evaluasi adalah sistem ini dirancang untuk digunakan oleh pengguna dengan input berupa URL.

Selain ini perbedaannya terdapat pada proses ekstraksi yang sangat penting dalam menentukan kinerja sistem. Untuk lebih jelasnya dapat dilihat pada *flowchart* sistem berikut:



Gambar 3.2.2 Flowchart Sistem Implementasi

Sistem implementasi dalam deteksi *website phishing* ini dimulai dengan mengumpulkan dataset yang akan digunakan dalam proses pelatihan model. Data ini dikumpulkan dari berbagai sumber dan diperoleh dalam bentuk URL yang telah diklasifikasikan sebagai *phishing* atau legitimate. Setelah dataset diinputkan, maka data tersebut diolah dengan menggunakan algoritma *Gradient Boosting*. Algoritma ini dipilih karena dapat menangani

masalah klasifikasi dengan baik dan memiliki tingkat akurasi yang tinggi.

Setelah model pelatihan selesai, maka kita dapat menggunakan algoritma tersebut untuk digunakan saat membangun *website* yang bertujuan untuk mendeteksi *website phishing*. Dalam tahap ini, kita akan membuat sebuah form input yang digunakan untuk memasukkan URL yang akan diuji. Selanjutnya, kita akan melakukan ekstraksi fitur dari URL yang telah diinputkan. Fitur-fitur ini akan digunakan sebagai input dalam model yang telah dibuat sebelumnya.

Setelah fitur-fitur di ekstrak, maka data tersebut akan masuk kedalam model yang telah dibuat. Model ini akan menghasilkan sebuah *decision function* yang digunakan untuk menentukan apakah suatu *website* merupakan *website phishing* atau *legitimate*. *Decision function* ini akan mengevaluasi fitur-fitur yang telah diinputkan dan memberikan hasil dalam bentuk prediksi yang dapat digunakan untuk menentukan apakah suatu *website* merupakan *website phishing* atau *legitimate*. Proses deteksi *website phishing* akan selesai setelah *decision function* ini dijalankan dan hasilnya ditampilkan kepada pengguna.

3.3 Metodologi Data Science

Metodologi data science untuk proyek deteksi

halaman website phishing menggunakan algoritma Machine Learning Gradient Boosting Classifier adalah sebagai berikut:

1. Definisi masalah: Pemahaman akan masalah deteksi halaman website phishing dan tujuan dari proyek ini adalah untuk mengembangkan sebuah sistem yang dapat mendeteksi halaman website phishing dengan tingkat akurasi tinggi.
2. Pengumpulan data: Mengumpulkan data halaman website phishing dan legitimate dari berbagai sumber seperti database, file spreadsheet, atau melalui scraping dari website. Data ini digunakan untuk melatih dan menguji model.
3. Analisis data: Melakukan analisis data untuk mengekstrak fitur yang relevan dari setiap halaman website yang diambil. Fitur ini dapat meliputi informasi seperti URL, jumlah link internal dan eksternal, dan kata-kata yang digunakan dalam halaman website.
4. Pemodelan: Membuat model dengan menggunakan algoritma Machine Learning Gradient Boosting Classifier. Model ini dibangun dengan menggunakan data latih yang telah dikumpulkan sebelumnya dan digunakan untuk memprediksi apakah suatu halaman website

merupakan phishing atau legitimate.

5. Evaluasi: Evaluasi model yang dibuat dengan menguji model dengan menggunakan data uji yang telah dikumpulkan sebelumnya. Model di evaluasi dengan menggunakan metrik seperti akurasi, recall, dan precision.
6. Deployment: Implementasi model dalam sebuah aplikasi atau sistem yang digunakan oleh pengguna akhir. Aplikasi ini dapat digunakan untuk memasukkan URL yang akan diuji dan menampilkan hasil prediksi dari model.

BAB 4

Pembuatan Model Machine Learning

4.1 Import Library Tahap Awal

```
#import library yang diperlukan

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn import metrics
import warnings
warnings.filterwarnings('ignore')
```

Gambar 4.1 Import Library Tahap Awal

Pada *script* di atas, pertama-tama terdapat beberapa *library* yang akan diimpor ke dalam program. *Library* yang diimpor adalah:

1. *NumPy*: *Library* ini menyediakan fungsi-fungsi matematika yang berguna untuk melakukan operasi pada *array*.

Cara *install*: untuk menginstal *library numpy* anda dapat menginstallnya dengan menggunakan **Conda** dan **Pip**.

- Ketika menggunakan *conda* langkah pertama jalankan perintah-perintah berikut:

```
- # Best practice, use an environment
  rather than install in the base env

- conda create -n my-env

- conda activate my-env

- # If you want to install from conda-forge

- conda config --env --add channels conda-
  forge

- # The actual install command

- conda install numpy
```

- Ketika menggunakan Pip dapat menjalankan perintah berikut:

```
pip install numpy
```

2. *pandas*: Library ini memungkinkan kita untuk membaca dan mengelola data serta menyediakan berbagai fungsi untuk mengeksplorasi data.

Cara install:

- Untuk menginstall *pandas* kita bisa menggunakan perintah berikut:

```
pip install pandas
```

3. *Matplotlib*: Library ini memungkinkan kita untuk membuat berbagai jenis plot, seperti plot garis, plot bar, dan lain-lain.

Cara install: Untuk menginstall *matplotlib* kita dapat menggunakan 2 cara yaitu dengan menggunakan **Conda** dan **Pip**.

- Untuk pengguna conda dapat menjalankan perintah berikut:

Conda install matplotlib

Untuk penggunaan conda forge dapat menggunakan perintah berikut:

Conda install -c conda-forge matplotlib

- Untuk penggunaan via pip dapat menggunakan perintah berikut:

- **Python -m pip install -U pip**

- **Python -m pip install -U matplotlib**

4. *seaborn: Library* ini merupakan *library* yang berbasis pada *Matplotlib* dan menyediakan tipe plot yang lebih bervariasi serta lebih mudah digunakan.

Cara install: Untuk menginstall seaborn kita dapat menggunakan **Conda** dan **Pip**.

- Untuk penggunaan Conda dapat menjalankan perintah berikut:

- **Conda install seaborn**

- **Conda install seaborn -c conda-forge**

- Untuk penggunaan via Pip dapat menjalankan perintah berikut:

- **Pip install seaborn**

5. *sklearn: Library* ini merupakan *library* yang

menyediakan berbagai algoritma *machine learning* yang berguna untuk melakukan pembelajaran mesin. Cara *install*: Untuk cara penginstallan sklearn dapat menggunakan perintah berikut:

- `pip install -U scikit-learn`

Pada akhir *script*, terdapat perintah "`warnings.filterwarnings('ignore')`" yang berfungsi untuk menonaktifkan *warning* yang mungkin muncul saat menjalankan program.

4.2 Memuat Dataset

Sumber Dataset <https://www.kaggle.com/eswarchandt/phishing-website-detector>.

Kumpulan URL situs web untuk 11000+ situs web. Setiap sampel memiliki 30 parameter situs web dan label kelas yang mengidentifikasinya sebagai situs web *phishing* atau bukan (1 atau -1).

Gambaran umum dari dataset ini adalah, memiliki 11054 sampel dengan 32 fitur. Unduh dataset dari tautan yang disediakan.

Pertama, dataset ini menyediakan 11,054 URL situs web yang merupakan contoh dari situs web yang ada di internet. Setiap sampel dalam dataset ini mewakili sebuah situs web yang unik dan memiliki URL yang sesuai.

Kedua, setiap sampel dalam dataset ini juga memiliki label kelas yang mengidentifikasinya sebagai

situs web *phishing* atau bukan. Label kelas ini dapat berupa 1 atau -1, dimana 1 menandakan bahwa situs web tersebut adalah situs web *phishing*, sedangkan -1 menandakan bahwa situs web tersebut bukan situs web *phishing*.

Selain itu, dataset ini juga menyediakan 30 parameter situs web lainnya yang digunakan untuk mengidentifikasi situs web *phishing*. Parameter-parameter ini dapat berupa informasi seperti jumlah link dalam halaman web, jumlah form input, atau informasi lain yang dapat digunakan untuk mengidentifikasi situs web *phishing*.

Dalam penggunaan dataset ini, para peneliti atau pengembang dapat menggunakannya untuk membuat model yang dapat mengidentifikasi situs web *phishing* dengan menganalisis parameter-parameter yang tersedia. Selain itu, dataset ini juga dapat digunakan untuk mengevaluasi performa dari model yang sudah dibuat sebelumnya.

Secara keseluruhan, dataset ini dapat digunakan untuk meningkatkan keselamatan online dengan mengidentifikasi situs web *phishing* sebelum pengguna mengunjungi situs tersebut. Dataset ini bisa diunduh dari tautan yang disediakan dalam paragraf.


```
#Loading data into dataframe

data = pd.read_csv("phishing.csv")
data.head()
```

Gambar 4.2.1 Memuat Dataset

Hasil:

	Index	UsingIP	LongURL	ShortURL	Symbol@	Redirecting//	PrefixSuffix-	SubDomains	HTTPS	DomainRegLen	...	UsingPopupWindow	IframeRedirection
0	0	1	1	1	1	1	-1	0	1	-1	...	1	1
1	1	1	0	1	1	1	-1	-1	-1	-1	...	1	1
2	2	1	0	1	1	1	-1	-1	-1	1	...	1	1
3	3	1	0	-1	1	1	-1	1	1	-1	...	-1	1
4	4	-1	0	-1	1	-1	-1	1	1	-1	...	1	1

5 rows x 32 columns

Gambar 4.2.2 Hasil Memuat Dataset

4.3 Features Pada Data Set

Berikut ini merupakan penjelasan semua fitur yang terdapat didalam dataset:

- 1. **Index:** Menunjukkan apakah halaman website terindex oleh mesin pencari seperti Google atau tidak. Halaman *phishing* mungkin tidak terindex oleh mesin pencari.
- 2. **UsingIP:** Menunjukkan apakah halaman *website* menggunakan alamat IP atau nama domain. Halaman *phishing* mungkin menggunakan alamat IP.
- 3. **LongURL:** Menunjukkan panjang URL dari halaman *website*. Halaman *phishing* mungkin memiliki URL

yang panjang dan rumit.

4. **ShortURL**: Menunjukkan apakah halaman *website* menggunakan layanan URL pendek seperti bit.ly. Halaman phishing mungkin menggunakan layanan URL pendek untuk menyembunyikan alamat sebenarnya.
5. **Symbol@**: Menunjukkan apakah halaman *website* menggunakan simbol @ dalam URL. Halaman phishing mungkin menggunakan simbol @ dalam URL untuk menyembunyikan alamat sebenarnya.
6. **Redirecting//**: Menunjukkan apakah halaman *website* mengarahkan ke URL lain. Halaman phishing mungkin mengarahkan ke URL lain untuk menipu pengguna.
7. **PrefixSuffix-**: Menunjukkan apakah halaman *website* menggunakan prefiks atau sufiks dalam URL. Halaman *phishing* mungkin menggunakan *prefiks* atau *sufiks* dalam URL untuk menyembunyikan alamat sebenarnya.
8. **SubDomains**: Menunjukkan apakah halaman *website* menggunakan subdomain. Halaman *phishing* mungkin menggunakan subdomain untuk menipu pengguna.
9. **HTTPS**: Menunjukkan apakah halaman *website* menggunakan protokol HTTPS atau tidak. Halaman *phishing* mungkin tidak menggunakan HTTPS.

10. **DomainRegLen**: Menunjukkan berapa lama domain *website* terdaftar. Halaman *phishing* mungkin memiliki domain yang baru terdaftar.
11. **Favicon**: Menunjukkan apakah halaman *website* memiliki *favicon* atau tidak. Halaman *phishing* mungkin tidak memiliki *favicon*.
12. **NonStdPort**: Menunjukkan apakah halaman *website* menggunakan *port non-standar*. Halaman *phishing* mungkin menggunakan *port non-standar* untuk menyembunyikan alamat sebenarnya.
13. **HTTPSDomainURL**: Menunjukkan apakah alamat *website* menggunakan protokol HTTPS atau tidak. Halaman *phishing* mungkin tidak menggunakan HTTPS.
14. **RequestURL**: Menunjukkan apakah halaman *website* mengirim permintaan ke server. Halaman *phishing* mungkin mengirim permintaan yang tidak diinginkan ke server.
15. **AnchorURL**: Menunjukkan apakah halaman *website* mengandung *link-link* yang dapat mengarahkan pengguna ke halaman lain. Halaman *phishing* mungkin mengandung *link-link* yang dapat mengarahkan pengguna ke halaman *phishing* lainnya.
16. **LinksInScriptTags**: Menunjukkan apakah halaman *website* mengandung *link* yang tersembunyi di

dalam *tag script*. Halaman *phishing* mungkin mengandung *link* yang tersembunyi di dalam *tag script* untuk menipu pengguna.

17. **ServerFormHandler**: Menunjukkan apakah halaman *website* mengandung *form* yang ditangani oleh server. Halaman *phishing* mungkin mengandung *form* yang ditangani oleh *server* untuk mengumpulkan informasi pengguna.
18. **InfoEmail**: Menunjukkan apakah halaman *website* menyediakan informasi kontak *email*. Halaman *phishing* mungkin tidak menyediakan informasi kontak *email*.
19. **AbnormalURL**: Menunjukkan apakah halaman *website* memiliki URL yang tidak biasa. Halaman *phishing* mungkin memiliki URL yang tidak biasa untuk menyembunyikan alamat sebenarnya.
20. **WebsiteForwarding**: Menunjukkan apakah halaman *website* mengarahkan pengguna ke halaman lain. Halaman *phishing* mungkin mengarahkan pengguna ke halaman *phishing* lainnya.
21. **StatusBarCust**: Menunjukkan apakah halaman *website* mengubah tampilan status bar pada browser. Halaman *phishing* mungkin mengubah tampilan *status* bar untuk menipu pengguna.
22. **DisableRightClick**: Menunjukkan apakah halaman *website* menonaktifkan klik kanan pada *mouse*.

Halaman *phishing* mungkin menonaktifkan *klik* kanan untuk mencegah pengguna mengecek alamat sebenarnya.

23. **UsingPopupWindow:** Menunjukkan apakah halaman *website* menggunakan jendela *popup*. Halaman *phishing* mungkin menggunakan jendela *popup* untuk menipu pengguna.
24. **IframeRedirection:** Menunjukkan apakah halaman *website* mengarahkan pengguna ke halaman lain melalui *iframe*. Halaman *phishing* mungkin mengarahkan pengguna ke halaman *phishing* melalui *iframe*.
25. **AgeofDomain:** Menunjukkan berapa lama domain *website* didaftarkan. Halaman *phishing* mungkin memiliki domain yang baru didaftarkan.
26. **DNSRecording:** Menunjukkan apakah *domain website* memiliki catatan DNS. Halaman *phishing* mungkin tidak memiliki catatan DNS.
27. **WebsiteTraffic:** Menunjukkan jumlah *traffic* yang datang ke *website*. Halaman *phishing* mungkin memiliki *traffic* yang rendah.
28. **PageRank:** Menunjukkan nilai *PageRank website* dari Google. Halaman *phishing* mungkin memiliki nilai *PageRank* yang rendah.
29. **GoogleIndex:** Menunjukkan apakah halaman *website* terindeks oleh Google atau tidak. Halaman

phishing mungkin tidak terindeks oleh Google.

30. ***LinksPointingToPage***: Menunjukkan jumlah link yang mengarah ke halaman *website*. Halaman *phishing* mungkin memiliki jumlah link yang rendah.
31. ***StatsReport***: Menunjukkan laporan statistik dari halaman *website* seperti jumlah pengunjung, *bounce rate*, dan lama kunjungan. Halaman *phishing* mungkin memiliki laporan statistik yang tidak normal.
32. ***class*** : Menunjukkan apakah suatu halaman *website* merupakan *phishing* atau *legitimate*, yang merupakan hasil prediksi dari algoritma *machine learning* yang digunakan.

Semua fitur di atas digunakan sebagai input dalam algoritma *machine learning* yang digunakan untuk menentukan apakah suatu halaman *website* merupakan *phishing* atau *legitimate*.

4.4 Familiar dengan Data & EDA

EDA merupakan singkatan dari *Exploratory Data Analysis* (Analisis Data Eksploratif). EDA merupakan suatu proses yang dilakukan untuk mengeksplorasi data dengan tujuan untuk menemukan pola-pola yang terdapat dalam data tersebut.

```
#Shape of dataframe

data.shape

(11054, 32)
```

Gambar 4.3.1 Data Shape

Fungsi **shape** adalah suatu atribut yang dimiliki oleh object **DataFrame** yang terdapat dalam library pandas. Atribut ini mengembalikan tuple yang menunjukkan dimensi dari data, yaitu jumlah baris dan jumlah kolom.

```
#Listing the features of the dataset

data.columns

Index(['Index', 'UsingIP', 'LongURL', 'ShortURL', 'Symbol@', 'Redirecting//',
       'PrefixSuffix-', 'SubDomains', 'HTTPS', 'DomainRegLen', 'Favicon',
       'NonStdPort', 'HTTPSDomainURL', 'RequestURL', 'AnchorURL',
       'LinksInScriptTags', 'ServerFormHandler', 'InfoEmail', 'AbnormalURL',
       'WebsiteForwarding', 'StatusBarCust', 'DisableRightClick',
       'UsingPopupWindow', 'IframeRedirection', 'AgeofDomain', 'DNSRecording',
       'WebsiteTraffic', 'PageRank', 'GoogleIndex', 'LinksPointingToPage',
       'StatsReport', 'class'],
      dtype='object')
```

Gambar 4.3.2 Data Columns

Fungsi **columns** adalah suatu atribut yang dimiliki oleh object **DataFrame** yang terdapat dalam library pandas. Atribut ini mengembalikan nama-nama kolom dari data yang tersimpan dalam object **DataFrame** tersebut.

```
#Information about the dataset

data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11054 entries, 0 to 11053
Data columns (total 32 columns):
#   Column                Non-Null Count  Dtype
---  ---                ---
0   Index                 11054 non-null  int64
1   UsingIP               11054 non-null  int64
2   LongURL               11054 non-null  int64
3   ShortURL              11054 non-null  int64
4   Symbol@              11054 non-null  int64
5   Redirecting//         11054 non-null  int64
6   PrefixSuffix-         11054 non-null  int64
7   SubDomains            11054 non-null  int64
8   HTTPS                 11054 non-null  int64
9   DomainRegLen          11054 non-null  int64
10  Favicon               11054 non-null  int64
11  NonStdPort            11054 non-null  int64
12  HTTPSDomainURL        11054 non-null  int64
13  RequestURL            11054 non-null  int64
14  AnchorURL             11054 non-null  int64
15  LinksInScriptTags     11054 non-null  int64
16  ServerFormHandler     11054 non-null  int64
17  InfoEmail             11054 non-null  int64
18  AbnormalURL           11054 non-null  int64
19  WebsiteForwarding     11054 non-null  int64
20  StatusBarCust         11054 non-null  int64
```

Gambar 4.3.3 Data Info

Fungsi *info* adalah suatu method yang dimiliki oleh *object DataFrame* yang terdapat dalam *library* pandas. *Method* ini digunakan untuk menampilkan informasi mengenai tipe data, jumlah baris, dan nama-nama kolom yang terdapat dalam data.


```

#dropping index column
data = data.drop(['Index'],axis = 1)

#description of dataset
data.describe().T

```

	count	mean	std	min	25%	50%	75%	max
UsingIP	11054.0	0.313914	0.949495	-1.0	-1.0	1.0	1.0	1.0
LongURL	11054.0	-0.633345	0.765973	-1.0	-1.0	-1.0	-1.0	1.0
ShortURL	11054.0	0.738737	0.674024	-1.0	1.0	1.0	1.0	1.0
Symbol@	11054.0	0.700561	0.713625	-1.0	1.0	1.0	1.0	1.0
Redirecting//	11054.0	0.741632	0.670837	-1.0	1.0	1.0	1.0	1.0
PrefixSuffix-	11054.0	-0.734938	0.678165	-1.0	-1.0	-1.0	-1.0	1.0
SubDomains	11054.0	0.064049	0.817492	-1.0	-1.0	0.0	1.0	1.0
HTTPS	11054.0	0.251040	0.911856	-1.0	-1.0	1.0	1.0	1.0
DomainRegLen	11054.0	-0.336711	0.941651	-1.0	-1.0	-1.0	1.0	1.0
Favicon	11054.0	0.628551	0.777804	-1.0	1.0	1.0	1.0	1.0
NonStdPort	11054.0	0.728243	0.685350	-1.0	1.0	1.0	1.0	1.0
HTTPSDomainURL	11054.0	0.675231	0.737640	-1.0	1.0	1.0	1.0	1.0
RequestURI	11054.0	0.186720	0.982458	-1.0	-1.0	1.0	1.0	1.0

Gambar 4.3.4 Data Info

Fungsi *describe* adalah suatu method yang dimiliki oleh object **DataFrame** yang terdapat dalam library pandas. Method ini digunakan untuk menampilkan statistik deskriptif dari data, seperti mean, std, min, max, dan quartile.

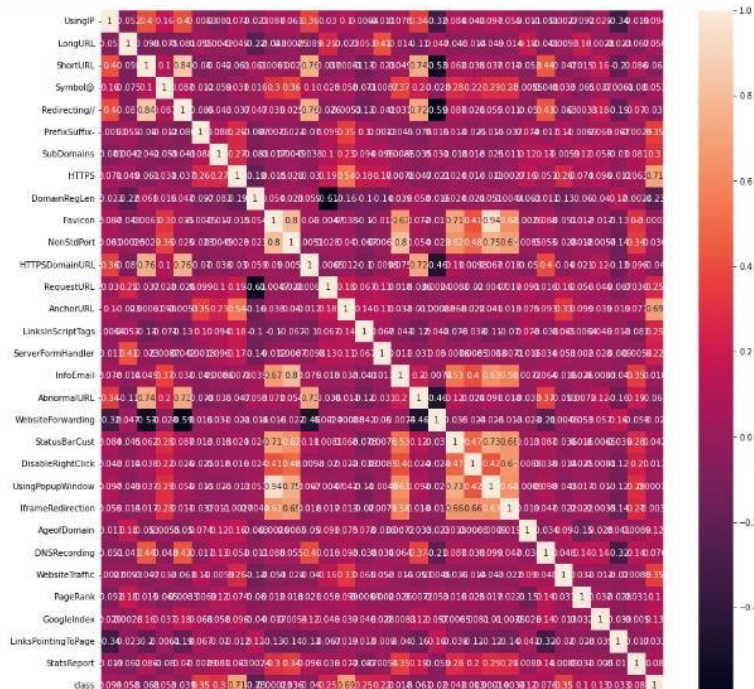
Method **T** merupakan method yang dimiliki oleh object DataFrame yang digunakan untuk men-transpose tabel, yaitu menukar posisi baris dan kolom.

4.5 Memvisualisasikan Data

```
#Correlation heatmap  
  
plt.figure(figsize=(15,15))  
sns.heatmap(data.corr(), annot=True)  
plt.show()
```

Gambar 4.4.1 Correlation heatmap

Pada *script* di atas, pertama-tama dilakukan inisialisasi *figure* dengan ukuran 15x15 inch menggunakan perintah **plt.figure(figsize=(15,15))**. Kemudian, dilakukan plot heatmap dengan menggunakan perintah **sns.heatmap()**. Argumen yang diberikan kepada perintah ini adalah matriks korelasi dari data (**data.corr()**) serta argumen **annot=True** yang digunakan untuk menampilkan nilai korelasi pada setiap sel heatmap. Terakhir, plot ditampilkan dengan perintah **plt.show()**.



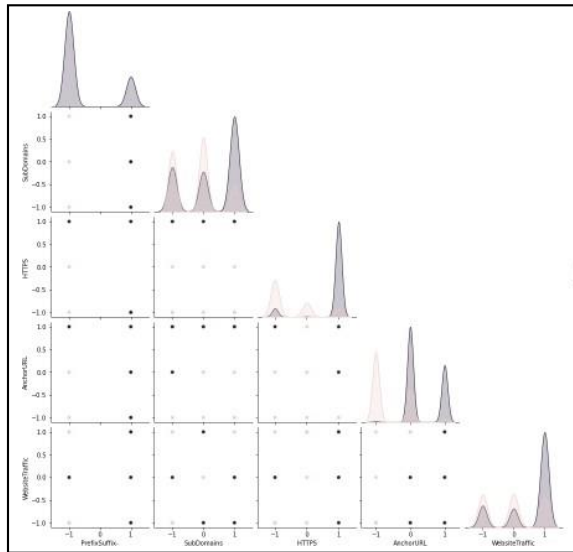
Gambar 4.4.2 Hasil Correlation heatmap

```
#pairplot untuk fitur tertentu
df = data[['PrefixSuffix-', 'SubDomains', 'HTTPS', 'AnchorURL', 'WebsiteTraffic', 'class']]
sns.pairplot(data = df, hue="class", corner=True);
```

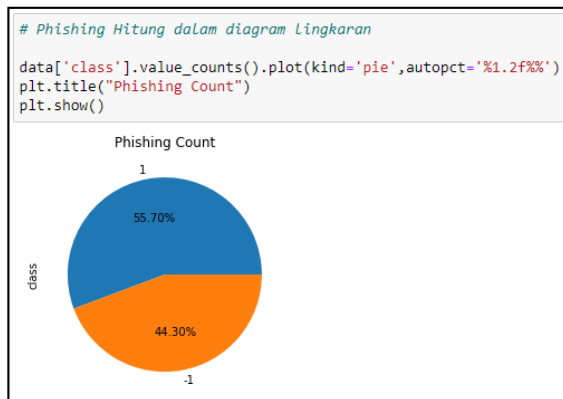
Gambar 4.4.3 Pairplot

Pada script di atas, pertama-tama dilakukan pemilihan fitur yang akan dianalisis dengan menggunakan notasi *indexing* pada *object DataFrame*. Kemudian, dilakukan plot pairplot dengan menggunakan perintah *sns.pairplot()*. Argumen yang diberikan kepada perintah ini adalah data yang akan dianalisis (*data = df*), kolom yang akan digunakan sebagai hue (*hue="class"*), dan

argumen ***corner=True*** yang digunakan untuk menampilkan plot diagonal yang menunjukkan distribusi data pada setiap fitur.



Gambar 4.4.4 Hasil Pairplot



Gambar 4.4.4 Diagram Lingkaran

Pada script di atas, pertama-tama dilakukan

penghitungan jumlah kelas yang terdapat dalam data dengan menggunakan method ***value_counts()***. Kemudian, dilakukan plot diagram lingkaran dengan menggunakan perintah ***plot(kind='pie')***. Argumen ***autopct='%1.2f%%'*** digunakan untuk menampilkan nilai persentase dari setiap kelas pada diagram lingkaran. Terakhir, judul plot ditambahkan dengan perintah ***plt.title()*** dan plot ditampilkan dengan perintah ***plt.show()***.

4.6 Memisahkan Dataset

```
# Memisahkan dataset menjadi fitur dependen dan independen

X = data.drop(["class"],axis =1)
y = data["class"]

# Membagi dataset menjadi set train dan test: pembagian 80-20

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
X_train.shape, y_train.shape, X_test.shape, y_test.shape

((8843, 30), (8843,), (2211, 30), (2211,))
```

Gambar 4.4.5 Memisahkan Dataset

Pada *script* di atas, pertama-tama dilakukan *import module* ***train_test_split*** dari *library* ***sklearn. model_selection.*** Kemudian, *module* tersebut digunakan untuk membagi dataset menjadi *set train* dan *set test* dengan perintah ***train_test_split(X, y, test_size = 0.2, random_state = 42)***. Argumen *X* dan *y* merupakan fitur dan label data, ***test_size*** merupakan proporsi data yang akan digunakan sebagai set test (dalam hal ini 0.2 atau 20%), dan ***random_state*** merupakan *seed* yang digunakan untuk memastikan bahwa *set train* dan *set test*

yang dihasilkan tidak berubah-ubah setiap kali *script* dijalankan.

Setelah dataset terbagi, maka akan dihasilkan 4 *object* yaitu ***X_train*** dan ***y_train*** yang merupakan *set train*, serta ***X_test*** dan ***y_test*** yang merupakan *set test*. Kemudian, *object* tersebut ditampilkan dengan perintah ***X_train.shape***, ***y_train.shape***, ***X_test.shape***, dan ***y_test.shape*** untuk menampilkan dimensi dari setiap *object*.

Pembagian dataset menjadi *set train* dan *set test* merupakan langkah yang penting dalam proses pembuatan model *machine learning*. *Set train* digunakan untuk melatih model sedangkan *set test* digunakan untuk mengevaluasi model yang telah dilatih. Dengan membagi dataset menjadi *set train* dan *set test*, kita dapat memastikan bahwa model yang dihasilkan tidak hanya dapat menangani data yang telah diketahui tapi juga dapat menangani data baru yang belum pernah dilihat sebelumnya.

4.7 Membangun dan Melatih Model

Supervised adalah salah satu jenis pembelajaran mesin yang paling umum digunakan dan berhasil. Pembelajaran yang *supervised* digunakan kapan pun ingin memprediksi hasil/label tertentu dari sekumpulan fitur tertentu, dan memiliki contoh pasangan fitur- label. Membangun model pembelajaran mesin dari

pasangan fitur-label ini, yang terdiri dari set train. Tujuannya adalah membuat prediksi akurat untuk data baru.

Ada dua jenis utama masalah supervised machine learning, yang disebut klasifikasi dan regresi. Kumpulan data berada di bawah masalah regresi. Model pembelajaran mesin supervised (regresi) yang dianggap melatih kumpulan data di notebook ini adalah:

1. *Logistic Regression*
2. *k-Nearest Neighbors*
3. *Support Vector Classifier*
4. *Naive Bayes*
5. *Decision Tree*
6. *Random Forest*
7. *Gradient Boosting*
8. *Catboost*
9. *Multilayer Perceptrons*

Metrik yang dipertimbangkan untuk mengevaluasi performa model adalah *Accuracy & F1 score*.

```

# Membuat holder untuk menyimpan hasil performa model
ML_Model = []
accuracy = []
f1_score = []
recall = []
precision = []

#fungsi untuk memanggil untuk menyimpan hasil
def storeResults(model, a,b,c,d):
    ML_Model.append(model)
    accuracy.append(round(a, 3))
    f1_score.append(round(b, 3))
    recall.append(round(c, 3))
    precision.append(round(d, 3))

```

Gambar 4.6 Membuat Holder Model

Pada *script* di atas, pertama-tama dibuat empat list yaitu ***ML_Model***, ***accuracy***, ***f1_score***, ***recall***, dan ***precision*** yang akan digunakan untuk menyimpan hasil performa model *machine learning*. Kemudian, dibuat sebuah fungsi ***storeResults()*** yang memiliki empat argumen yaitu ***model***, a, b, c, dan d. Fungsi ini akan menyimpan argumen argumen tersebut ke dalam masing-masing list yang telah dibuat sebelumnya.

Fungsi ***storeResults()*** ini akan berguna untuk menyimpan hasil performa model machine learning yang diuji. Dengan menyimpan hasil performa tersebut, kita dapat membandingkan performa model yang berbeda serta menentukan model terbaik yang akan digunakan pada data.

4.7.1 Model Logistic Regression

```
# Linear regression model
from sklearn.linear_model import LogisticRegression
#from sklearn.pipeline import Pipeline

# instantiate the model
log = LogisticRegression()

# fit the model
log.fit(X_train,y_train)

LogisticRegression()
```

Gambar 4.6.1 Model Logistic Regression

Script di atas memuat sebuah model *regresi logistik* yang akan dilatih menggunakan data latih. Terdapat beberapa bagian yang terdapat pada *script* di atas, yaitu:

1. Import kelas ***LogisticRegression*** dari modul ***sklearn.linear_model***. Kelas ***LogisticRegression*** merupakan kelas yang digunakan untuk membuat model regresi *logistik*.
2. Instansiasi objek model dengan menggunakan kelas ***LogisticRegression***. Objek model ini akan dibuat dengan menggunakan *default parameter* yang telah ditentukan oleh kelas ***LogisticRegression***.
3. Melatih model dengan menggunakan data latih dengan memanggil fungsi ***fit*** pada objek model yang telah dibuat. Fungsi ***fit*** akan melatih model dengan menggunakan data latih yang diberikan.

Setelah model dilatih, kita dapat menggunakannya untuk memprediksi target pada data uji dengan memanggil fungsi ***predict*** pada objek model yang telah dilatih. Selanjutnya, kita dapat mengevaluasi performa model dengan menghitung beberapa metrik seperti akurasi, *f1-score*, *recall*, *precision* dan mulai membuatnya.

4.7.2 Model k-Nearest Neighbors

```
# K-Nearest Neighbors Classifier model
from sklearn.neighbors import KNeighborsClassifier

# instantiate the model
knn = KNeighborsClassifier(n_neighbors=1)

# fit the model
knn.fit(X_train,y_train)
```

Gambar 4.6.2 Model KNN

Script di atas memuat sebuah model *K-Nearest Neighbors* (KNN) yang akan dilatih menggunakan data latih. Terdapat beberapa bagian yang terdapat pada script di atas, yaitu:

1. Import kelas ***KNeighborsClassifier*** dari modul *sklearn.neighbors*. Kelas ***KNeighborsClassifier*** merupakan kelas yang digunakan untuk membuat model KNN.
2. Instansiasi objek model dengan menggunakan kelas ***KNeighborsClassifier***. Objek model ini akan

dibuat dengan menggunakan parameter *n_neighbors=1*, yang menandakan bahwa model KNN yang akan dibuat akan menggunakan 1 tetangga terdekat dari setiap titik data.

3. Melatih model dengan menggunakan data latih dengan memanggil fungsi *fit* pada objek model yang telah dibuat. Fungsi *fit* akan melatih model dengan menggunakan data latih yang diberikan. Setelah model dilatih, kita dapat menggunakannya untuk memprediksi target pada data uji dengan memanggil fungsi *predict* pada objek model yang telah dilatih. Selanjutnya, kita dapat mengevaluasi performa model dengan menghitung beberapa metrik seperti akurasi, *f1-score*, *recall*, *precision* dan mulai membuat scriptnya.

4.7.3 Model SVM

```
# Support Vector Classifier model
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV

# defining parameter range
param_grid = {'gamma': [0.1], 'kernel': ['rbf', 'linear']}

svc = GridSearchCV(SVC(), param_grid)

# fitting the model for grid search
svc.fit(X_train, y_train)
```

Gambar 4.6.3 Model SVM

Script di atas memuat sebuah model SVM yang akan dilatih menggunakan data latih. Terdapat beberapa bagian yang terdapat pada *script* di atas, yaitu:

1. Import kelas **SVC** dari modul ***sklearn.svm***. Kelas **SVC** merupakan kelas yang digunakan untuk membuat model SVM.
2. Instansiasi objek model dengan menggunakan kelas **SVC** Objek model ini akan dibuat dengan menggunakan parameter ***param grid***.
3. Melatih model dengan menggunakan data latih dengan memanggil fungsi ***fit*** pada objek model yang telah dibuat. Fungsi ***fit*** akan melatih model dengan menggunakan data latih yang diberikan. Setelah model dilatih, kita dapat menggunakannya

untuk memprediksi target pada data uji dengan memanggil fungsi **predict** pada objek model yang telah dilatih. Selanjutnya, kita dapat mengevaluasi performa model dengan menghitung beberapa metrik seperti akurasi, *f1-score*, *recall*, *precision* dan mulai membuaascriptnya.

4.7.4 Model Naïve Bayes

```
# Naive Bayes Classifier Model
from sklearn.naive_bayes import GaussianNB
from sklearn.pipeline import Pipeline

# instantiate the model
nb= GaussianNB()

# fit the model
nb.fit(X_train,y_train)

GaussianNB()
```

Gambar 4.6.4 Model Naïve Bayes

Script di atas mengimport kelas *GaussianNB* dari library *scikit-learn* yang digunakan untuk membuat model *Naive Bayes Classifier* dengan asumsi bahwa fitur-fitur dari data telah ditransformasikan menjadi distribusi *Gaussian*. Kemudian, kelas *Pipeline* juga diimport untuk mengelompokkan transformasi data dan pemodelan ke dalam satu tahap.

Setelah itu, sebuah objek dari kelas *GaussianNB* dibuat dengan menggunakan perintah "**nb = *GaussianNB*()**". Kemudian, model tersebut di-fit dengan data latih menggunakan perintah "**nb.fit(X_train, y_train)**". *X_train* adalah variabel yang berisi data fitur untuk data latih, sedangkan *y_train* adalah variabel yang berisi label untuk data latih. Dengan memanggil method *fit()*, model *Naive Bayes Classifier* akan dilatih dengan mempelajari korelasi antara fitur-fitur dari data dan label yang sesuai.

4.7.5 Model Decision Tree

```
# Decision Tree Classifier model
from sklearn.tree import DecisionTreeClassifier

# instantiate the model
tree = DecisionTreeClassifier(max_depth=30)

# fit the model
tree.fit(X_train, y_train)

DecisionTreeClassifier(max_depth=30)
```

Gambar 4.6.5 Model Decision Tree

Script di atas mengimport kelas *DecisionTreeClassifier* dari library *scikit-learn* yang digunakan untuk membuat model *Decision Tree Classifier*. Kemudian, sebuah objek dari kelas *DecisionTreeClassifier* dibuat dengan menggunakan perintah "**tree = DecisionTreeClassifier (max_depth=30)**". Perintah ini akan membuat sebuah *model Decision Tree Classifier* dengan depth maksimum 30. *Depth* maksimum adalah jumlah maksimum dari node dari pohon keputusan.

Setelah itu, model tersebut di-fit dengan data latih menggunakan perintah "**tree.fit(X_train, y_train)**". *X_train* adalah variabel yang berisi data fitur untuk data latih, sedangkan *y_train* adalah variabel yang berisi label untuk data latih. Dengan memanggil method *fit()*, model *Decision Tree Classifier* akan dilatih dengan mempelajari korelasi antara fitur-fitur dari data dan label yang sesuai,

kemudian menggunakan informasi tersebut untuk membuat sebuah pohon keputusan yang akan digunakan untuk melakukan klasifikasi.

4.7.6 Model Random Forest

```
# Random Forest Classifier Model
from sklearn.ensemble import RandomForestClassifier

# instantiate the model
forest = RandomForestClassifier(n_estimators=10)

# fit the model
forest.fit(X_train,y_train)

RandomForestClassifier(n_estimators=10)
```

Gambar 4.6.6 Model Random Forest

Script di atas mengimport kelas ***RandomForestClassifier*** dari library *scikit-learn* yang digunakan untuk membuat model *Random Forest Classifier*. Kemudian, sebuah objek dari kelas *RandomForestClassifier* dibuat dengan menggunakan perintah "***forest = RandomForestClassifier (n_estimators= 10)***". Perintah ini akan membuat sebuah model *Random Forest Classifier* dengan menggunakan sebanyak 10 pohon keputusan yang dibuat secara acak.

Setelah itu, model tersebut di-fit dengan data latih menggunakan perintah "***forest.fit(X_train, y_train)***". *X_train* adalah variabel yang berisi data fitur untuk data latih, sedangkan *y_train* adalah variabel yang berisi label untuk data latih. Dengan memanggil *method* *fit()*, model

Random Forest Classifier akan dilatih dengan mempelajari korelasi antara fitur-fitur dari data dan label yang sesuai, kemudian menggunakan informasi tersebut untuk membuat sebanyak 10 pohon keputusan yang dibuat secara acak. Kemudian, hasil dari setiap pohon keputusan tersebut akan digabungkan untuk memprediksi label dari data baru.

4.7.7 Model Gradient Boosting

```
# Gradient Boosting Classifier Model
from sklearn.ensemble import GradientBoostingClassifier

# instantiate the model
gbc = GradientBoostingClassifier(max_depth=4, learning_rate=0.7)

# fit the model
gbc.fit(X_train, y_train)

GradientBoostingClassifier(learning_rate=0.7, max_depth=4)
```

Gambar 4.6.7 Model Gradient Boosting

Script di atas mengimport kelas ***GradientBoostingClassifier*** dari library *scikit-learn* yang digunakan untuk membuat model *Gradient Boosting Classifier*. Kemudian, sebuah objek dari kelas *GradientBoostingClassifier* dibuat dengan menggunakan perintah "***gbc = GradientBoostingClassifier (max_depth=4, learning_rate=0.7)***". Perintah ini akan membuat sebuah model *Gradient Boosting Classifier* dengan depth maksimum 4 dan learning rate 0,7. Depth maksimum adalah jumlah maksimum dari node dari pohon keputusan, sedangkan learning rate adalah

seberapa cepat model belajar dari data.

Setelah itu, model tersebut di-fit dengan data latih menggunakan perintah "*gbc.fit(X_train, y_train)*". *X_train* adalah variabel yang berisi data fitur untuk data latih, sedangkan *y_train* adalah variabel yang berisi label untuk data latih. Dengan memanggil method *fit()*, model *Gradient Boosting Classifier* akan dilatih dengan mempelajari korelasi antara fitur-fitur dari data dan label yang sesuai, kemudian menggunakan informasi tersebut untuk membuat pohon keputusan yang akan digunakan untuk melakukan klasifikasi. Model ini akan terus memperbaiki hasilnya dengan menambahkan pohon keputusan baru sesuai dengan hasil dari pohon keputusan sebelumnya.

4.7.8 Model Catboost

```
# catboost Classifier Model
from catboost import CatBoostClassifier

# instantiate the model
cat = CatBoostClassifier(learning_rate = 0.1)

# fit the model
cat.fit(X_train,y_train)
```

Gambar 4.6.8 Model Catboost

Script di atas mengimport kelas *CatBoostClassifier* dari library *catboost* yang digunakan untuk membuat model *CatBoost Classifier*. *CatBoost* adalah sebuah algoritma *machine learning* yang dapat digunakan untuk

melakukan klasifikasi, regresi, dan klasterisasi. Kemudian, sebuah objek dari kelas *CatBoostClassifier* dibuat dengan menggunakan perintah **"cat = CatBoostClassifier(learning_rate=0.1)"**. Perintah ini akan membuat sebuah model *CatBoost Classifier* dengan learning rate 0,1. *Learning* rate adalah seberapa cepat model belajar dari data.

Setelah itu, model tersebut di-fit dengan data latih menggunakan perintah **"cat.fit(X_train, y_train)"**. *X_train* adalah variabel yang berisi data fitur untuk data latih, sedangkan *y_train* adalah variabel yang berisi label untuk data latih. Dengan memanggil method *fit()*, model *CatBoost Classifier* akan dilatih dengan mempelajari korelasi antara fitur-fitur dari data dan label yang sesuai, kemudian menggunakan informasi tersebut untuk membuat pohon keputusan yang akan digunakan untuk melakukan klasifikasi. Model ini juga menggunakan teknik "*gradient boosting*" yang akan terus memperbaiki hasilnya dengan menambahkan pohon keputusan baru sesuai dengan hasil dari pohon keputusan sebelumnya.

4.7.9 Model Multilayer Perceptrons

```
# Multi-Layer Perceptron Classifier Model
from sklearn.neural_network import MLPClassifier

# instantiate the model
mlp = MLPClassifier()
#mlp = GridSearchCV(mlpc, parameter_space)

# fit the model
mlp.fit(X_train,y_train)

MLPClassifier()
```

Gambar 4.6.8 Model Multilayer Perceptrons

Script di atas mengimport kelas ***MLPClassifier*** dari library *scikit-learn* yang digunakan untuk membuat model ***Multi-layer Perceptron Classifier***. MLP (*Multi-layer Perceptron*) adalah sebuah jenis jaringan syaraf tiruan yang terdiri dari satu atau lebih lapisan tersembunyi yang menghubungkan input dan output. Kemudian, sebuah objek dari kelas ***MLPClassifier*** dibuat dengan menggunakan perintah "***mlp = MLPClassifier()***". Perintah ini akan membuat sebuah model ***Multi-layer Perceptron Classifier*** dengan parameter default.

Setelah itu, model tersebut di-fit dengan data latih menggunakan perintah "***mlp.fit(X_train, y_train)***". *X_train* adalah variabel yang berisi data fitur untuk data latih, sedangkan *y_train* adalah variabel yang berisi label untuk data latih. Dengan memanggil method *fit()*, model ***Multi-layer Perceptron Classifier*** akan dilatih dengan mempelajari korelasi antara fitur-fitur dari data dan label

yang sesuai, kemudian menggunakan informasi tersebut untuk membuat sebuah jaringan syaraf tiruan yang akan digunakan untuk melakukan klasifikasi.

Baris kode *"mlp = GridSearchCV(mlpc, parameter_space)"* tidak aktif dan tidak akan dijalankan. Baris kode ini akan mencari parameter terbaik untuk model *Multi-layer Perceptron Classifier* dengan menggunakan algoritma *Grid Search*. *Parameter_space* adalah *dictionary* yang berisi daftar parameter yang akan diuji.

4.8 Pendandingan Model

<pre>#creating dataframe result = pd.DataFrame({ 'ML Model' : ML_Model, 'Accuracy' : accuracy, 'f1_score' : f1_score, 'Recall' : recall, 'Precision': precision, })</pre>					
<pre># displaying total result result</pre>					
	ML Model	Accuracy	f1_score	Recall	Precision
0	Logistic Regression	0.934	0.941	0.943	0.927
1	K-Nearest Neighbors	0.956	0.961	0.991	0.989
2	Support Vector Machine	0.964	0.968	0.980	0.965
3	Naive Bayes Classifier	0.605	0.454	0.292	0.997
4	Decision Tree	0.957	0.962	0.991	0.993
5	Random Forest	0.965	0.969	0.993	0.990
6	Gradient Boosting Classifier	0.974	0.977	0.994	0.986
7	CatBoost Classifier	0.972	0.975	0.994	0.989
8	Multi-layer Perceptron	0.968	0.972	0.996	0.977

Gambar 4.7.1 Membuat Dataframe Perbandingan Model

Pada script di atas, dibuat sebuah *object result* yang merupakan sebuah *DataFrame* dengan

menggunakan *library pandas*. **DataFrame** dibuat dengan menggunakan argumen yang berisi *dictionary* dengan *key* sebagai nama kolom dan *value* sebagai isi dari kolom tersebut.

DataFrame merupakan tipe data yang terdiri dari baris dan kolom yang mirip dengan tabel pada *database*. **DataFrame** dapat dibuat dengan menggunakan data yang berasal dari file atau dengan memasukkan data secara manual seperti yang telah dilakukan pada *script* di atas.

Dalam kasus ini, **DataFrame** yang dihasilkan akan berisi informasi mengenai nama model *machine learning*, akurasi, *f1-score*, *recall*, dan *precision*. Informasi tersebut akan disimpan dalam masing-masing kolom yang telah didefinisikan sebelumnya. **DataFrame** ini akan berguna untuk menyimpan dan menampilkan hasil performa model *machine learning* dengan lebih mudah.

```
#Sorting the dataframe on accuracy
sorted_result=result.sort_values(by=['Accuracy', 'f1_score'],ascending=False).reset_index(drop=True)

# displaying total result
sorted_result
```

	ML Model	Accuracy	f1_score	Recall	Precision
0	Gradient Boosting Classifier	0.974	0.977	0.994	0.986
1	CatBoost Classifier	0.972	0.975	0.994	0.989
2	Multi-layer Perceptron	0.968	0.972	0.996	0.977
3	Random Forest	0.965	0.969	0.993	0.990
4	Support Vector Machine	0.964	0.968	0.980	0.965
5	Decision Tree	0.957	0.962	0.991	0.993
6	K-Nearest Neighbors	0.956	0.961	0.991	0.989
7	Logistic Regression	0.934	0.941	0.943	0.927
8	Naive Bayes Classifier	0.605	0.454	0.292	0.997

Gambar 4.7.2 Sorting Dataframe Accuracy

Dalam kasus ini, **DataFrame** *result* akan disortir

berdasarkan kolom *Accuracy* dan *f1_score* dengan urutan *descending* (nilai yang lebih tinggi lebih dulu). Kemudian, *method reset_index()* dengan argumen *drop=True* digunakan untuk menghilangkan index yang lama dan menggantinya dengan index yang baru yang diurutkan sesuai dengan urutan data.

Hasil dari *sorting* tersebut disimpan ke dalam *object* baru yaitu *sorted_result*. *Sorting* pada *DataFrame* dapat berguna untuk menyusun data sesuai dengan prioritas yang diinginkan sehingga lebih mudah untuk dianalisis.

4.9 Menyimpan Model Terbaik

```
# XGBoost Classifier Model
from xgboost import XGBClassifier

# instantiate the model
gbc = GradientBoostingClassifier(max_depth=4, learning_rate=0.7)

# fit the model
gbc.fit(X_train, y_train)

GradientBoostingClassifier(learning_rate=0.7, max_depth=4)
```

Gambar 4.8 Menyimpan Model Terbaik

Pada script di atas, pertama-tama dilakukan *import library XGBClassifier* dari *library xgboost*. Kemudian, dilakukan instansiasi object *gbc* dengan menggunakan *class XGBClassifier*. Argumen yang diberikan kepada class tersebut adalah *max_depth* yang merupakan jumlah maksimum dari tingkat dalam pohon model, dan *learning_rate* yang merupakan tingkat pembelajaran dari

model.

Setelah *object gbc* terbuat, dilakukan *training* model dengan menggunakan *method fit()*. *Method* ini membutuhkan dua argumen yaitu *X_train* dan *y_train* yang merupakan *set train* yang telah dibuat sebelumnya. Setelah *training* selesai, maka model yang telah dilatih akan tersimpan dalam *object gbc*.

4.10 Ubah Menjadi Format Pickle

```
import pickle

# dump information to that file
pickle.dump(gbc, open('model1.pkl', 'wb'))
```

Gambar 4.9.1 Import Library pickle

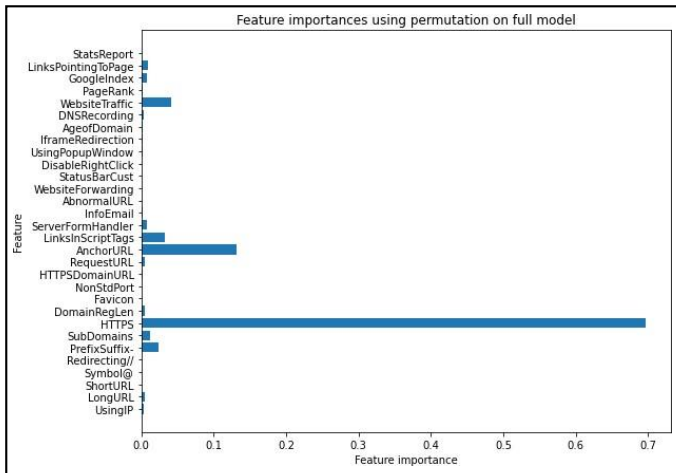
Pada script di atas, dilakukan *import library pickle*. Kemudian, dilakukan proses serialisasi dengan menggunakan *method dump()* dari *library pickle*. *Method* ini membutuhkan dua argumen yaitu object yang akan diserialisasi (dalam hal ini *object gbc* yang merupakan model yang telah dilatih) dan file yang akan digunakan untuk menyimpan hasil serialisasi (dalam hal ini file **model1.pkl**).

```
#checking the feature importance in the model
plt.figure(figsize=(9,7))
n_features = X_train.shape[1]
plt.barh(range(n_features), gbc.feature_importances_, align='center')
plt.yticks(np.arange(n_features), X_train.columns)
plt.title("Feature importances using permutation on full model")
plt.xlabel("Feature importance")
plt.ylabel("Feature")
plt.show()
```

Gambar 4.9.2 Cek Fitur pada Model

Pada script di atas, dilakukan *plot bar* dengan menggunakan *library matplotlib*. *Figure* yang dibuat akan memiliki ukuran 9x7. Kemudian, dihitung jumlah fitur yang terdapat dalam *set train* dengan ***X_train.shape[1]*** dan dibuat *bar* dengan *method barh()* yang memiliki argumen ***range(n_features)*** sebagai posisi sumbu x, ***gbc.feature_importances_*** sebagai lebar bar, dan ***align='center'*** sebagai posisi sumbu y. Kemudian, ditambahkan label pada sumbu y dengan menggunakan *method yticks()* dan menampilkan nama-nama fitur yang terdapat dalam set train dengan ***X_train.columns***. Terakhir, ditambahkan judul plot dengan *method title()*, label sumbu x dengan *method xlabel()*, dan label sumbu y dengan *method ylabel()*. Plot tersebut ditampilkan dengan *method show()*.

Gambar 4.9.3 Hasil Cek Fitur pada Model



4.11 Penelitian Sebelumnya

- a. **Peneliti:** Pungkas Subarkah, Ali Nur Ikhsan

Judul: *IDENTIFIKASI WEBSITE PHISHING MENGGUNAKAN ALGORITMA CLASSIFICATION AND REGRESSION TREES (CART)*

Hasil Penelitian: Berdasarkan penelitian yang telah dilakukan mengenai identifikasi *website phishing* menggunakan Algoritma *Classification And Regression Trees (CART)* dapat disimpulkan bahwa dari nilai *confusion matrix* diperoleh hasil akurasi sebesar 95.28%, dengan rincian nilai *precision* sebesar 0.953%, nilai *recall* sebesar 0.953% dan nilai *F-Measure* sebesar 0.953%. Dari hasil nilai akurasi dikategorikan klasifikasi sangat baik.[18]

- b. **Peneliti:** Diki Wahyudi, Muhammad Niswar, Ais Prayogi Alimuddin

Judul: *Website Phising Detection Application Using Support Vector Machine (SVM)*

Hasil Penelitian: Didapat hasil *Decision Tree* dengan nilai *accuracy* 85.40%. [19]

BAB 5

Pembuatan Aplikasi Web Phising

Pembuatan Aplikasi Web dengan *Flask*. *Flask* adalah sebuah *micro web framework* yang ditulis dalam bahasa pemrograman *Python*. *Framework* ini memiliki struktur yang sederhana dan mudah digunakan, sehingga cocok bagi pemula yang ingin belajar membuat aplikasi web.

Pembuatan aplikasi web dengan *Flask* dapat dilakukan dengan beberapa langkah sederhana. Pertama, pastikan bahwa *Python* dan *Flask* sudah terinstall pada komputer Anda. Jika belum, Anda dapat menginstallnya dengan perintah "*pip install flask*" pada *command prompt* atau terminal. Kemudian, buat sebuah file *Python* baru dan tambahkan baris kode berikut:

Gambar 5.1 Script Flask

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def home():
    return "Hello, World!"

if __name__ == '__main__':
    app.run()
```

5.1 Install Flask

Untuk menginstall *Flask*, pertama-tama pastikan bahwa *Python* sudah terinstall pada komputer Anda. Jika sudah, Anda dapat mengikuti langkah-langkah berikut:

1. Buka *command prompt* atau terminal Anda.
2. Aktifkan *virtual environment* jika Anda menggunakannya. Jika belum memiliki *virtual environment*, Anda dapat membuatnya dengan perintah "*python -m venv nama_virtual_environment*".
3. *Install Flask* dengan perintah "*pip install flask*". Ini akan menginstall *Flask* dan semua dependensi yang diperlukan.
4. Jika ingin mengecek apakah *Flask* sudah terinstall dengan benar, buat sebuah file *Python* baru dan tambahkan baris kode berikut:

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello():
    return "Hello, Flask!"

if __name__ == '__main__':
    app.run()
```

Gambar 5.1.1 Cek Flask Sudah Berjalan

5. Jalankan file tersebut dengan perintah "*python*

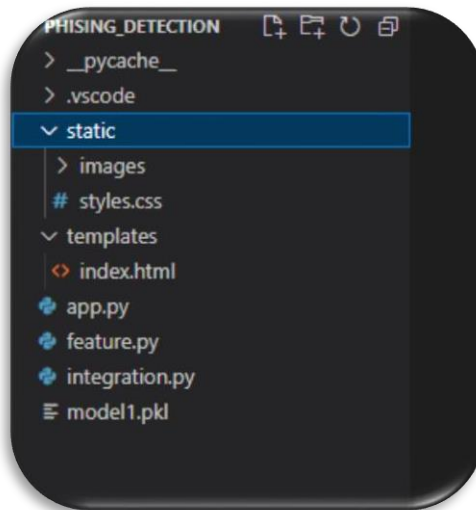
nama_file.py". Jika *Flask* sudah terinstall dengan benar, Anda akan mendapatkan output "*Running on http://127.0.0.1:5000/*".

6. Buka browser Anda dan akses alamat tersebut. Anda akan melihat output "Hello, Flask!" pada halaman web.

5.2 Membuat Struktur Folder

Pada bagian ini kita akan membuat struktur folder yang sederhana untuk membuat aplikasi web dimana aplikasi ini nantinya akan menampilkan halaman website pendeteksi *phishing*. Berikut merupakan gambaran struktur sederhana dalam pembuat aplikasi *website* pendeteksi *phishing* ini.

Gambar 5.2.1 Struktur Folder Flask



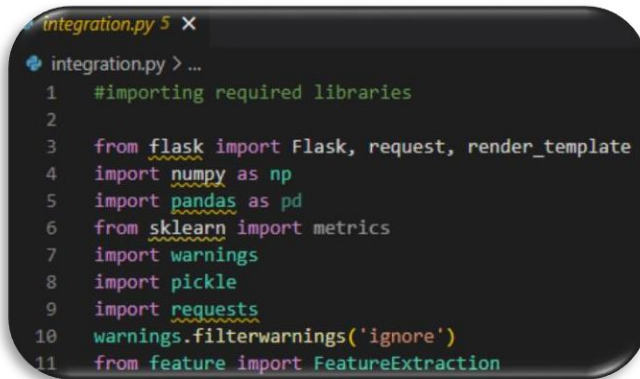
5.3 Import Pickle

Silahkan perhatikan kembali pada struktur folder

diatas, disana terdapat **modell1.pkl**, Dimana model tersebut merupakan hasil *convert* model yang kita buat sebelumnya di *jupyter notebook*. Kita akan menggunakan atau mengkonsumsi model tersebut untuk dapat menentukan web phishing berdasarkan model yang kita buat sebelumnya dan kita tentukan yaitu *Gradient Boosting Classifier*.

Selanjutnya kita akan bermain-main dengan file **integration.py**, dimana file ini akan digunakan untuk dapat menggunakan model yang sudah kita simpan kedalam pickle. Berikut merupakan script”nya:

1. *Import* terlebih dahulu untuk *library* yang digunakan untuk keperluan penggunaan model.



```
integration.py 5 X
integration.py > ...
1  #importing required libraries
2
3  from flask import Flask, request, render_template
4  import numpy as np
5  import pandas as pd
6  from sklearn import metrics
7  import warnings
8  import pickle
9  import requests
10 warnings.filterwarnings('ignore')
11 from feature import FeatureExtraction
```

Gambar 5.3.1 Import Library Flask

2. Lalu panggil *modell1.pkl* dan lakukan load untuk filenya.

```
12
13  file = open("model1.pkl", "rb")
14  gbc = pickle.load(file)
15  file.close()
16
17  app = Flask(__name__)
18
```

Gambar 5.3.2 Panggil Model

Script di atas membuka file bernama "**model1.pkl**" dengan mode 'rb' (*read binary*). File ini diasumsikan berisi objek yang telah disimpan menggunakan *pickle*. Kemudian, objek tersebut dimuat menggunakan *method pickle.load()* dan disimpan ke dalam variabel *gbc*. Setelah itu, file ditutup dengan menggunakan *method close()*.

Setelah objek dimuat, sebuah objek dari kelas *Flask* juga dibuat dengan menggunakan perintah "**app = Flask(name)**". Objek ini akan digunakan untuk membuat aplikasi web dengan *Flask*.

3. Selanjutnya buatlah script untuk dapat mengekstraksi hasil yang telah dihasilkan saat membuat model.

```

19 @app.route("/", methods=["GET", "POST"])
20 def index():
21     if request.method == "POST":
22
23         url = request.form["url"]
24         obj = FeatureExtraction(url)
25         x = np.array(obj.getFeaturesList()).reshape(1,30)
26
27         y_pred = gbc.predict(x)[0]
28         #1 is safe
29         #-1 is unsafe
30         y_pro_phishing = gbc.predict_proba(x)[0,0]
31         y_pro_non_phishing = gbc.predict_proba(x)[0,1]
32         # If(y_pred ==1.):
33         pred = "It is {:.2f} % safe to go ".format(y_pro_phishing*100)
34         payload_scoring = {"input_data": [{"field": [{"UsingIP", "LongURL", "ShortURL", "Symbol@", "Redirecting//",
35             }, "values": [[1,1,1,1,1,-1,-1,-1,-1,1,1,1,1,-1,-1,1,1,1,0,1,1,1,-1,-1,-1,-1,1,0,1]]]}]}
36         # headers={'Authorization': 'Bearer ' + accessToken)
37         response_scoring = requests.post(' ', json=payload_scoring)
38         print("Scoring response")
39         predictions=response_scoring.json()

```

Gambar 5.3.3 Ekstraksi Model

4. Kirim Hasilnya kedalam template *index.html* dengan menggunakan *render*, dan akses via web *browser*.

Setelah melakukan proses pengolahan data dan pembuatan model, hasil dari proses tersebut harus di tampilkan kepada pengguna agar dapat digunakan dengan baik. Salah satu cara untuk menampilkan hasil tersebut adalah dengan menggunakan *render* dan akses melalui web *browser*.

Render adalah proses untuk mengubah data mentah menjadi tampilan yang dapat diterima oleh pengguna. Dalam hal ini, hasil dari proses pengolahan data dan pembuatan model akan diubah menjadi tampilan yang dapat diterima oleh web *browser*. Dengan menggunakan *render*, data mentah yang tidak dapat dibaca oleh pengguna akan diubah menjadi tampilan yang mudah dipahami.

Setelah hasil dari proses pengolahan data dan pembuatan model diubah menjadi tampilan yang dapat diterima oleh web *browser*, hasil tersebut dapat diakses melalui web *browser*. Web *browser* adalah perangkat yang digunakan untuk mengakses internet dan menampilkan halaman web. Pengguna dapat mengakses hasil dari proses pengolahan data dan pembuatan model dengan mengetikkan alamat URL yang sesuai pada web *browser*.

Secara keseluruhan, dengan menggunakan *render* dan akses melalui web *browser*, hasil dari proses pengolahan data dan pembuatan model akan dapat diakses oleh pengguna dengan mudah dan dapat dipahami dengan baik. *Render* akan mengubah data mentah menjadi tampilan yang dapat diterima oleh pengguna dan web *browser* akan menampilkan hasil tersebut kepada pengguna. Dan hasil tersebut dapat diinputkan kedalam template *index.html* yang sudah disediakan, sehingga user dapat dengan mudah mengakses dan memahami hasil yang diperoleh dari proses pengolahan data dan pembuatan model.


```

41     pred=print(predictions['predictions'][0]['values'][0][0])
42     return render_template('index.html',xx =round(y_pro_non_phishing,2),url=url )
43     return render_template("index.html", xx =-1)
44
45
46 if __name__ == "__main__":
47     app.run(debug=True,port=2020)

```

Gambar 5.3.4 Render Model

5.4 Hasil

Setelah selesai melakukan proses pengolahan data dan pembuatan model, selanjutnya adalah melakukan pengecekan URL pada sistem deteksi *phishing*. Langkah pertama yang harus dilakukan adalah menjalankan aplikasi *Flask*.

Aplikasi *Flask* adalah sebuah *framework* yang digunakan untuk membuat aplikasi web dengan *Python*. Aplikasi ini sangat fleksibel dan mudah digunakan, sehingga cocok digunakan dalam pengembangan sistem deteksi *phishing*. Setelah menjalankan aplikasi *Flask*, pengguna akan diarahkan ke halaman utama aplikasi.

Di halaman utama aplikasi, pengguna akan dapat melihat tombol merah yang dapat diklik. Tombol ini akan mengarahkan pengguna ke bagian cek URL. Bagian ini merupakan bagian yang paling penting dari sistem deteksi *phishing*.

Di bagian cek URL, pengguna dapat memasukkan URL yang ingin dicek. Setelah URL dimasukkan, sistem akan menganalisis URL tersebut menggunakan model

yang telah dibuat sebelumnya. Model ini akan mengevaluasi URL tersebut dan memberikan hasil yang dapat diterima oleh pengguna. Hasil dari analisis ini dapat berupa situs web phishing atau bukan situs web *phishing*.

Secara keseluruhan, dengan menjalankan aplikasi *Flask* dan mengklik tombol merah di halaman utama, pengguna akan diarahkan ke bagian cek URL yang merupakan bagian terpenting dari sistem deteksi phishing. Di bagian ini, pengguna dapat memasukkan URL yang ingin dicek dan sistem akan mengevaluasi URL tersebut menggunakan model yang telah dibuat sebelumnya. Hasil dari analisis ini akan memberikan informasi yang dapat digunakan oleh pengguna untuk mengambil tindakan yang tepat.

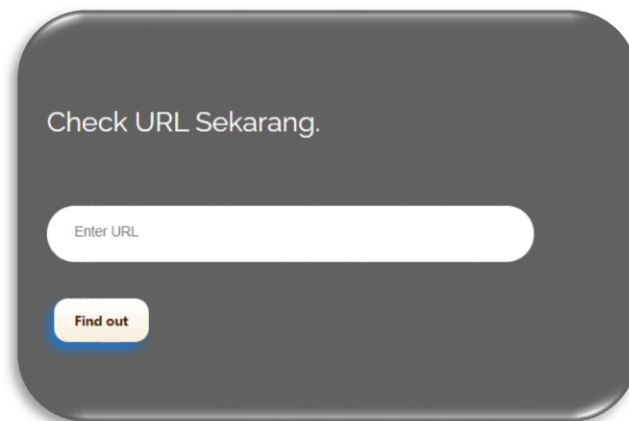
Gambar 5.4.1 Halaman Web Awal



Setelah diarahkan ke bagian cek URL, pengguna dapat melakukan pengecekan terhadap suatu URL yang ingin dianalisis. Hal pertama yang harus dilakukan adalah

menyisipkan atau mengetikkan URL yang akan dianalisis pada kolom yang tersedia. Pengguna dapat menyisipkan URL dengan menggunakan cara *copy paste* atau dengan mengetikkan URL secara manual.

Setelah URL dimasukkan, pengguna dapat mengklik tombol "*find out*" untuk memulai proses analisis. Tombol ini akan mengaktifkan sistem untuk melakukan analisis terhadap URL yang telah dimasukkan. Setelah analisis selesai, sistem akan menampilkan hasil dari analisis tersebut. Hasil dari analisis ini dapat berupa situs web *phishing* atau bukan situs web *phishing*.



Gambar 5.4.2 Cek URL

Sebagai contoh pertama, kita akan mencoba untuk memasukkan URL <https://atacado-barato-magazine.myshopify.com/> pada kolom pengecekan yang tersedia. Setelah URL dimasukkan, kita akan mengklik tombol "*find*

out" untuk memulai proses analisis.

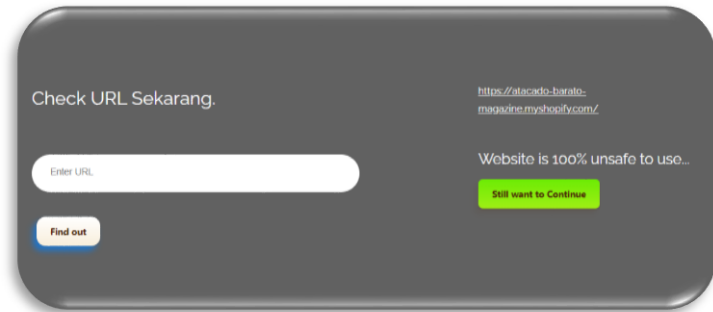
Proses analisis akan dilakukan dengan menggunakan model yang telah dibuat sebelumnya. Model ini akan menganalisis setiap parameter yang terkait dengan URL yang dimasukkan dan mengevaluasi URL tersebut untuk menentukan apakah URL tersebut merupakan situs web *phishing* atau bukan.

Setelah proses analisis selesai, sistem akan menampilkan hasil dari analisis tersebut. Pada contoh ini, sistem mendeteksi bahwa situs yang dimasukkan merupakan situs web *phishing*. Ini berarti bahwa URL yang dimasukkan merupakan situs web yang tidak dapat dipercaya dan dapat menyebabkan kerugian pada pengguna jika dikunjungi.

Pengguna harus mengambil tindakan yang tepat untuk menghindari kerugian yang mungkin terjadi. Tindakan yang dapat diambil antara lain seperti menghindari untuk mengklik *link* atau mengisi form yang ada pada situs tersebut. Selain itu, sebaiknya juga melaporkan situs tersebut kepada pihak yang berwenang untuk mengambil tindakan lebih lanjut.

Secara keseluruhan, dengan menggunakan sistem deteksi *phishing*, pengguna dapat mengetahui apakah suatu URL merupakan situs web *phishing* atau bukan. Ini dapat membantu pengguna untuk mengambil tindakan yang tepat untuk menghindari kerugian yang mungkin

terjadi.



Gambar 5.4.3 Hasil URL Phishing

Selanjutnya, kita akan mencoba untuk melakukan pengecekan kedua dengan menggunakan contoh website dari Medium yaitu <https://medium.com>. Sama seperti pada contoh sebelumnya, kita akan memasukkan URL tersebut pada kolom pengecekan dan mengklik tombol "*find out*" untuk memulai proses analisis.

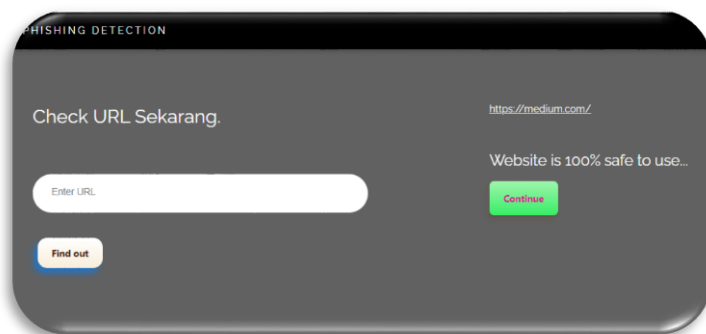
Proses analisis akan dilakukan dengan menggunakan model yang telah dibuat sebelumnya. Model ini akan menganalisis setiap parameter yang terkait dengan URL yang dimasukkan dan mengevaluasi URL tersebut untuk menentukan apakah URL tersebut merupakan situs web *phishing* atau bukan.

Setelah proses analisis selesai, sistem akan menampilkan hasil dari analisis tersebut. Pada contoh ini, sistem menyatakan bahwa *website* medium yang diinputkan baik untuk diakses. Ini berarti bahwa URL yang dimasukkan merupakan situs web yang dapat dipercaya

dan aman untuk dikunjungi.

Hal ini sangat penting karena dengan mengetahui *website* yang aman, pengguna dapat mengakses *website* tersebut tanpa khawatir akan terjadi kerugian. Selain itu, pengguna juga dapat mengambil tindakan yang tepat untuk menghindari kerugian yang mungkin terjadi, seperti menghindari untuk mengklik *link* atau mengisi *form* yang ada pada situs yang tidak aman.

Secara keseluruhan, dengan menggunakan sistem deteksi *phishing*, pengguna dapat mengetahui apakah suatu URL merupakan situs web *phishing* atau bukan. Ini dapat membantu pengguna untuk mengambil tindakan yang tepat untuk menghindari kerugian yang mungkin terjadi dan mengakses *website* yang aman.



Gambar 5.4.4 Hasil URL tidak Phishing

5.5 Kesimpulan

Secara keseluruhan, sistem deteksi *phishing* adalah alat yang penting untuk meningkatkan

keselamatan *online*. Dengan menggunakan sistem ini, pengguna dapat mengetahui apakah suatu URL merupakan situs web *phishing* atau bukan sebelum mengunjungi situs tersebut. Ini dapat membantu pengguna untuk mengambil tindakan yang tepat untuk menghindari kerugian yang mungkin terjadi dan mengakses *website* yang aman.

Sistem ini dapat dijalankan dengan menggunakan *Flask*, sebuah *framework* yang digunakan untuk membuat aplikasi web dengan *Python*. Aplikasi ini sangat fleksibel dan mudah digunakan, sehingga cocok digunakan dalam pengembangan sistem deteksi *phishing*. Setelah menjalankan aplikasi *Flask*, pengguna dapat mengakses bagian cek URL dengan mengklik tombol yang disediakan, di mana pengguna dapat memasukkan URL yang ingin dicek dan sistem akan mengevaluasi URL tersebut menggunakan model yang telah dibuat sebelumnya.

DAFTAR PUSTAKA

- [1] Wahyudi Diki, Niswar Muhammad. 2022. “*Website Phising Detection Application Using Vektor Machine (SVM)*”. Journal of Information Technology and Its Utilization, Volume 5 : 18.
- [2]Deekshitha B., Aswitha Ch, Dkk, 2022. “*URL Based Phishing Website Detection by Using Gradient and Catboost Algorithms*”.*International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, Volume 10.
- [3]Deekshitha B., Aswitha Ch, Dkk, 2022. “*URL Based Phishing Website Detection by Using Gradient and Catboost Algorithms*”.*International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, Volume 10.
- [4] Halim Z. 2017. “*Prediksi Website Pemancing Informasi Penting Phising Menggunakan Support Vector Machine (SVM)*”. Information System for Educators and Professionals. 2 (1): 71 – 82.
- [5]Sadli Muhammad, Fajriana, Fuadi Wahyu, 2018, Dkk. “*Penerapan Model K-Nearest Neighbors Dalam Klasifikasi Kebutuhan Daya Listrik Untuk masing-masing Daerah di Kota Lhokseumawe*”. Jurnal ECOTIPE, Volume 5, No.2.
- [6] Altaher Taha Altyeb, 2017. “*Phishing Websites Classification using Hybrid SVM and KNN Approach*”. (IJACSA) International Journal of Advanced Computer Science and Applications, Vol.8, No.6.
- [7]Kumar Narander, Chaudhary Priyanka, 2017. “*Mobile Phishing*

- Detection using Naive Bayesian Algorithm*". IJCSNS International Journal of Computer Science and Network Security, VOL.17 No.7
- [8] Mao Jian, Bian Jingdong Bian, Tian Wenqian, Dkk, 2019. "*Phishing page detection via learning classifiers from page layout feature*". EURASIP Journal on Wileress Communication and Networking. No.43.
- [9]Kumar Dutta Ashit, 2021. "*Detecting phishing websites using machine learning technique*". Detecting phishing websites using machine learning technique. PLoS ONE 16(10): e0258361.
- [10]Deekshitha B., Aswitha Ch, Dkk, 2022. "*URL Based Phishing Website Detection by Using Gradient and Catboost Algorithms*". *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, Volume 10.
- [11] Chian Fang Lim, Ayop Zakiah, Anawar Syarulnaziah, Dkk, 2021. "*URL Phishing Detection System Utilizing Catboost Machine Learning Approach*". IJCSNS International Journal of Computer Science and Network Security, VOL.21 No.9.
- [12] Srinath, K. R, 2017. "*Python – The Fastest Growing Programming Language*". International Research Journal of Engineering and Technology (IRJET). Volume 4.
- [13] Irsyad Rahadian, "*Penggunaan Python Web Framework Flask Untuk Pemula*". Laboratorium Telematika, Sekolah Teknik Elektro & Informatika.
- [14] Setiabudidaya Dedi. "*PENGUNAAN PIRANTI LUNAK JUPYTER NOTEBOOK DALAM UPAYA MENSOSIALISASIKAN OPEN SCIENCE*". Universitas Sriwijaya, Jl. Raya PalembangPrabumulih KM 32, Indralaya 30662.

- [15] Permana Yudi, Romadlon Puji, 2019. “*PERANCANGAN SISTEM INFORMASI PENJUALAN PERUMAHAN MENGGUNAKAN METODE SDLC PADA PT. MANDIRI LAND PROSPEROUS BERBASIS MOBILE*”. Volume 10 Nomor 2 Desember 2019 ISSN : 2407-3903.
- [16] C. Price Danny, Van der Velden Ellert, Dkk, 2018. “*Hickle: A HDF5-based python pickle replacement*”. Journal of Open Source Software, 3(32), 1115.
- [17] Arif Alfis, Isro Yogi, Dkk, 2017. “*RANCANG BANGUN WEBSITE SEKOLAH MENENGAH PERTAMA (SMP) NEGERI 8 KOTA PAGARALAM*”. Jurnal Ilmiah Betrik, Vol. 08, No.03.
- [18] Subarkah Pungkas, Nur Ikhsan Ali, 2021. “*IDENTIFIKASI WEBSITE PHISHING MENGGUNAKAN ALGORITMA CLASSIFICATION AND REGRESSION TREES (CART)*”. Jurnal Ilmiah Informatika (Scientific Informatics Journal) with CC BY NC licence, P. Subarkah dkk/ JIMI 6 (2) pp. 127-136.
- [19] Wahyudi Diki, Niswar Muhammad. 2022. “*Website Phising Detection Application Using Vektor Machine (SVM)*”. Journal of Information Technology and Its Utilization, Volume 5 : 18.

-oo00oo-

PROFIL PENULIS

Nama : Muhammad Rizal Supriadi

Pendidikan : D4 Teknik Informatika

Perguruan Tinggi : Universitas Logistik dan Bisnis Internasional

Link Github: <https://github.com/KerjaBhakti/PhisingDetection>



DETEKSI HALAMAN WEBSITE PHISHING MENGGUNAKAN ALGORITMA MACHINE LEARNING GRADIENT BOOSTING CLASSIFIER

Phising merupakan suatu tindakan untuk mendapatkan informasi penting milik seseorang berupa username, password dan informasi penting lainnya dengan menyediakan situs web palsu yang mirip dengan aslinya. Phising adalah bentuk tindakan pidana yang bermaksud untuk mendapatkan informasi rahasia dari seseorang, seperti nama pengguna, sandi dengan menyamar sebagai orang atau bisnis terpercaya. Seiring dengan perkembangannya penggunaan media elektronik yang diikuti dengan meningkatnya kejahatan dunia maya, seperti serangan phising.

Oleh karena itu, untuk meminimalkan serangan phising, diperlukan sistem yang dapat mendeteksi serangan tersebut. Machine Learning merupakan salah satu metode yang dapat digunakan untuk membuat sistem yang dapat mendeteksi phising. Data yang digunakan dalam penelitian ini adalah 11054 data website yang terbagi menjadi dua bagian yaitu “sah” dan “phising”. Sedangkan algoritma yang digunakan adalah Gradient Boosting Classifier. Dari hasil pengujian pada penelitian ini diperoleh akurasi sistem terbaik 97,4% dengan nilai $f1_score$ 0,977 serta untuk aplikasi web menggunakan framework Flask bahasa pemrograman python.

Penerbit:

