A picture containing icon

Description automatically generatedA picture containing icon

Description automatically generatedA picture containing icon

Description automatically generated

**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

***Penulis :***

Muhammad Rizal Supriadi

Roni Andarsyah

ISBN : -

***Editor :***

Nisa Hanum Harani

***Penyunting :***

Nisa Hanum Harani

***Desain sampul dan Tata letak :***

Muhammad Rizal Supriadi

***Penerbit :***

Penerbit Buku Pedia

***Redaksi :***

Athena Residence Blok. E No. 1, Desa Ciwaruga,

Kec. Parongpong, Kab. Bandung Barat 40559

Tel. 628-775-2000-300

Email : penerbit@bukupedia.co.id

***Distributor :***

Informatics Research Center

Jl. Sariasih No. 54

Bandung 40151

Email : irc@poltekpos.ac.id

Cetakan Pertama, 2023

Hak cipta dilindungi undang-undang

Dilarang memperbanyak karya tulis ini dalam bentuk dan

dengan cara apa pun tanpa ijin tertulis dari penerbit

# KATA PENGANTAR

S

elamat 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 *phising*.

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

Bandung, 18 Oktober 2021

Penulis

Muhammad Rizal Supriadi

**DAFTAR ISI**

[KATA PENGANTAR i](#_Toc124287293)

[DAFTAR ISI ii](#_Toc124287294)

[DAFTAR GAMBAR v](#_Toc124287295)

[BAB 1](#_Toc124287296) [Pendahuluan 1](#_Toc124287297)

[1.1 Pengenalan Machine Learning 2](#_Toc124287298)

[1.2 Supervised Learning 4](#_Toc124287299)

[1.2.1 Logistic Regression 4](#_Toc124287300)

[1.2.2 K-Nearest Neighbors 6](#_Toc124287301)

[1.2.3 Support Vector Machine 8](#_Toc124287302)

[1.2.4 Naïve Bayes 10](#_Toc124287303)

[1.2.5 Decision Tree 12](#_Toc124287304)

[1.2.6 Random Forest 14](#_Toc124287305)

[1.2.7 Gradient Boosting 16](#_Toc124287306)

[1.2.8 Catboost 18](#_Toc124287307)

[1.3 Python 20](#_Toc124287308)

[BAB 2](#_Toc124287309) [Pengenalan Tools 21](#_Toc124287310)

[2.1 Framework Flask 21](#_Toc124287311)

[2.2 Jupyter Notebook 22](#_Toc124287312)

[2.3 Visual Studio Code 23](#_Toc124287313)

[2.4 Pickle 24](#_Toc124287314)

[2.5 Web Browser 24](#_Toc124287315)

[BAB 3](#_Toc124287316) [Metodologi Penelitian 26](#_Toc124287317)

[3.1 Diagram Alur Metodologi Penelitian 26](#_Toc124287318)

[3.2 Tahapan – tahapan Diagram Alur 28](#_Toc124287319)

[3.2.1 Sistem Evaluasi 28](#_Toc124287320)

[3.2.2 Sistem Implementasi 29](#_Toc124287321)

[3.3 Metodologi Data Science 28](#_Toc124287319)

[BAB 4](#_Toc124287322) [Pembuatan Model Machine Learning 34](#_Toc124287323)

[4.1 Import Library Tahap Awal 34](#_Toc124287324)

[4.2 Memuat Dataset 37](#_Toc124287325)

4.3 Features Pada Dataset 37

[4.3 Familiar dengan data & EDA 38](#_Toc124287326)

[4.4 Memvisualisasikan Data 46](#_Toc124287327)

[4.5 Mimisahkan Dataset 49](#_Toc124287328)

[4.6 Membangun dan Melatih Model 50](#_Toc124287329)

[4.6.1 Model Logistic Regression 52](#_Toc124287330)

[4.6.2 Model k-Nearest Neighbors 53](#_Toc124287331)

[4.6.3 Model SVM 54](#_Toc124287332)

[4.6.4 Model Naïve Bayes 55](#_Toc124287333)

[4.6.5 Model Decision Tree 56](#_Toc124287334)

[4.6.6 Model Random Forest 57](#_Toc124287335)

[4.6.7 Model Gradient Boosting 58](#_Toc124287336)

[4.6.8 Model Catboost 59](#_Toc124287337)

[4.6.9 Model Multilayer Perceptrons 60](#_Toc124287338)

[4.7 Pendandingan Model 61](#_Toc124287339)

[4.8 Menyimpan Model Terbaik 63](#_Toc124287340)

[4.9 Ubah Menjadi Format Pickle 63](#_Toc124287341)

[4.10 Penelitian Sebelumnya 65](#_Toc124287342)

[BAB 5](#_Toc124287343) [Pembuatan Aplikasi Web Phising 67](#_Toc124287344)

[5.1 Install Flask 68](#_Toc124287345)

[5.2 Membuat Struktur Folder 69](#_Toc124287346)

[5.3 Import Pickle 69](#_Toc124287347)

[5.4 Hasil 72](#_Toc124287348)

[5.5 Kesimpulan 72](#_Toc124287348)

[DAFTAR PUSTAKA 79](#_Toc124287349)

[GLOSARIUM 79](#_Toc124287349)

[INDEKS 79](#_Toc124287349)

[PROFIL 79](#_Toc124287349)

**DAFTAR GAMBAR**

[*Gambar 1.1 Machine Learning* 2](#_Toc124287293)

[*Gambar 1.2.1 Contoh Script Logistic Regression* 5](#_Toc124287293)

[*Gambar 1.2.2 Contoh Script KNN* 7](#_Toc124287293)

[*Gambar 1.2.3 Contoh Script SVM* 9](#_Toc124287293)

[*Gambar 1.2.4 Contoh Script Naïve Bayes* 11](#_Toc124287293)

[*Gambar 1.2.5 Contoh Script Decision Tree* 13](#_Toc124287293)

[*Gambar 1.2.6 Contoh Script Random Forest* 15](#_Toc124287293)

[*Gambar 1.2.7 Contoh Script Gradient Boosting* 17](#_Toc124287293)

[*Gambar 1.2.8 Contoh Script Catboost* 19](#_Toc124287293)

[*Gambar 2.1 Flask* 21](#_Toc124287293)

[*Gambar 2.2 Jupyter Notebook* 22](#_Toc124287293)

[*Gambar 2.3 VS Code* 23](#_Toc124287293)

[*Gambar 3.1 Diagram Alur Metodologi Penelitian* 26](#_Toc124287293)

[*Gambar 3.2.1 Flowchart Sistem Evaluasi* 29](#_Toc124287293)

[*Gambar 3.2.2 Flowchart Sistem Impelemtasi* 31](#_Toc124287293)

[*Gambar 4.1 Import Library Tahap Awal* 34](#_Toc124287293)

[*Gambar 4.2.1 Memuat Datase* 38](#_Toc124287293)

[*Gambar 4.2.2 Hasil Memuat Dataset* 38](#_Toc124287293)

[*Gambar 4.3.1 Data Shape* 42](#_Toc124287293)

[*Gambar 4.3.2 Data Columns* 43](#_Toc124287293)

[*Gambar 4.3.3 Data Info* 44](#_Toc124287293)

[*Gambar 4.3.4 Data Info* 45](#_Toc124287293)

[*Gambar 4.4.1 Correlation heatmap* 46](#_Toc124287293)

[*Gambar 4.4.2 Hasil Correlation heatmap* 47](#_Toc124287293)

[*Gambar 4.4.3 Pairplot* 47](#_Toc124287293)

[*Gambar 4.4.4 Hasil Pairplot* 48](#_Toc124287293)

[*Gambar 4.4.4 Diagram Lingkaran* 48](#_Toc124287293)

[*Gambar 4.4.5 Memisahkan Dataset* 49](#_Toc124287293)

[*Gambar 4.6 Membuat Holder Model* 51](#_Toc124287293)

[*Gambar 4.6.1 Model Logistic Regression* 52](#_Toc124287293)

[*Gambar 4.6.2 Model KNN* 53](#_Toc124287293)

[*Gambar 4.6.3 Model SVM* 54](#_Toc124287293)

[*Gambar 4.6.4 Model Naïve Bayes* 55](#_Toc124287293)

[*Gambar 4.6.5 Model Decision Tree* 56](#_Toc124287293)

[*Gambar 4.6.6 Model Random Forest* 57](#_Toc124287293)

[*Gambar 4.6.7 Model Gradient Boosting* 58](#_Toc124287293)

[*Gambar 4.6.8 Model Catboost* 59](#_Toc124287293)

[*Gambar 4.6.8 Model Multilayer Perceptrons* 60](#_Toc124287293)

[*Gambar 4.7.1 Membuat Dataframe Perbandingan Model* 61](#_Toc124287293)

[*Gambar 4.7.2 Sorting Dataframe Accuracy* 62](#_Toc124287293)

[*Gambar 4.8 Menyimpan Model Terbaik* 63](#_Toc124287293)

[*Gambar 4.9.1 Import Library pickle* 63](#_Toc124287293)

[*Gambar 4.9.2 Cek Fitur pada Model* 64](#_Toc124287293)

[*Gambar 4.9.3 Hasil Cek Fitur pada Model* 65](#_Toc124287293)

[*Gambar 5.1 Script Flask* 67](#_Toc124287293)

[*Gambar 5.1.1 Cek Flask Sudah Berjalan* 68](#_Toc124287293)

[*Gambar 5.2.1 Struktur Folder Flask* 69](#_Toc124287293)

[*Gambar 5.3.1 Import Library Flask* 70](#_Toc124287293)

[*Gambar 5.3.2 Panggil Model* 70](#_Toc124287293)

[*Gambar 5.3.3 Ekstraksi Model* 71](#_Toc124287293)

[*Gambar 5.3.4 Render Model* 72](#_Toc124287293)

[*Gambar 5.4.1 Halaman Web Awal* 74](#_Toc124287293)

[*Gambar 5.4.2 Check URL* 75](#_Toc124287293)

[*Gambar 5.4.3 Hasil URL Phishing* 76](#_Toc124287293)

[*Gambar 5.4.4 Hasil URL tidak Phishing* 77](#_Toc124287293)

# 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.

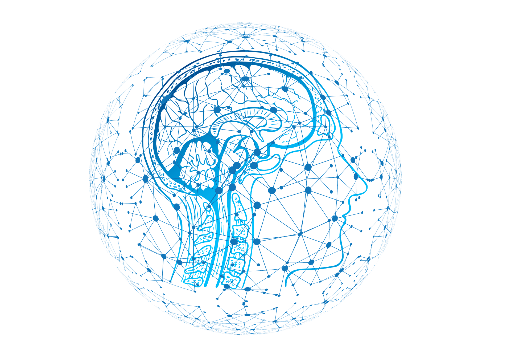
*Phishing* merupakan tindakan untuk mendapatkan informasi penting seseorang berupa *username, password* dan informasi sensitif lainnya dengan memberikan informasi palsu situs web yang mirip dengan aslinya. Pengelabuan (pemancing informasi penting) adalah sala satu bentuk tindak pidana yang bermaksud 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 Phising* 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 phising*, lalu melakukan *convert* model dan pembuatan aplikasi *website*.

* 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*:

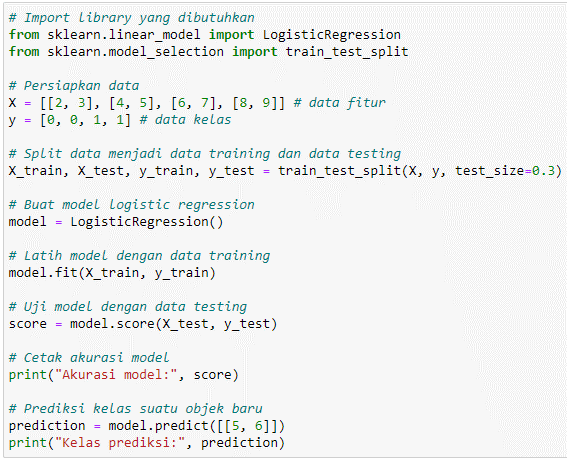
1. Sistem rekomendasi: Sistem rekomendasi menggunakan *machine learning* untuk mengelompokkan pengguna berdasarkan preferensi dan kebiasaan mereka, dan kemudian menyarankan produk atau layanan yang mungkin mereka sukai.
2. Analisis sentimen: *Machine learning* dapat digunakan untuk menganalisis sentimen 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. **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. **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*:



*Gambar 1.2.1 Contoh Script Logistic Regression*

Penjelasan *script* *Logistic Regression* diatas:

*Script* tersebut terdiri dari beberapa langkah:

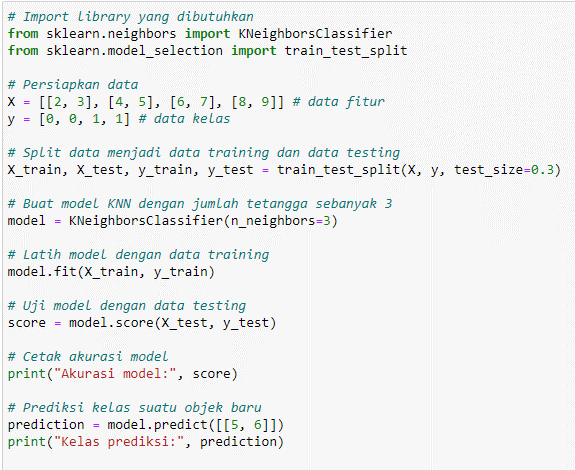
1. *Import library* yang dibutuhkan, yaitu ***LogisticRegression*** dan ***train\_test\_split*** dari *scikit-learn.*
2. Persiapkan data fitur dan kelas yang akan digunakan untuk melatih model.
3. *Split* data menjadi data training dan data *testing* dengan ukuran data *testing* sebesar 30%.
4. Buat 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. **K-Nearest Neighbors**

K-Nearest Neighboradalah metode klasifikasi dengan mencarijarak terdekat antara data yang akandievaluasi dengan *K-Nearest Neighbors*terdekatnya dalam data pelatihan. Modelini dapat digunakan dalam klasifikasiyang akan dilakukan data training didalamproses 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*:



*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.

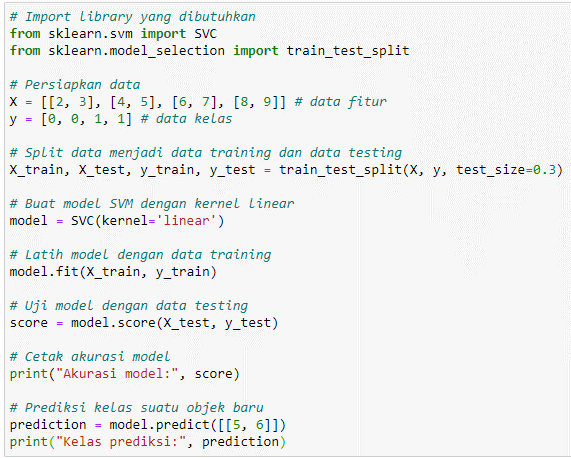
Setelah melakukan semua langkah tersebut, Anda akan mendapatkan model KNN yang telah dilatih dan diuji serta dapat digunakan untuk melakukan prediksi kelas suatu objek baru.

* + 1. **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*:



*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.

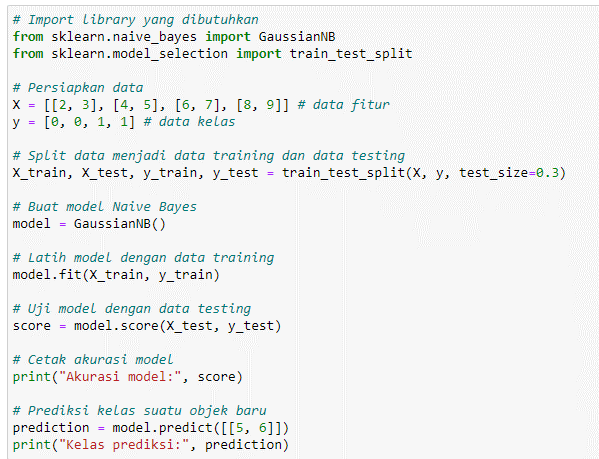
Setelah melakukan semua langkah tersebut, Anda akan mendapatkan model SVM yang telah dilatih dan diuji serta dapat digunakan untuk melakukan prediksi kelas suatu objek baru.

* + 1. **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*:



*Gambar 1.2.4 Contoh Script Naïve Bayes*

Penjelasan *script Naïve Bayes* diatas:

*Script* tersebut terdiri dari beberapa langkah:

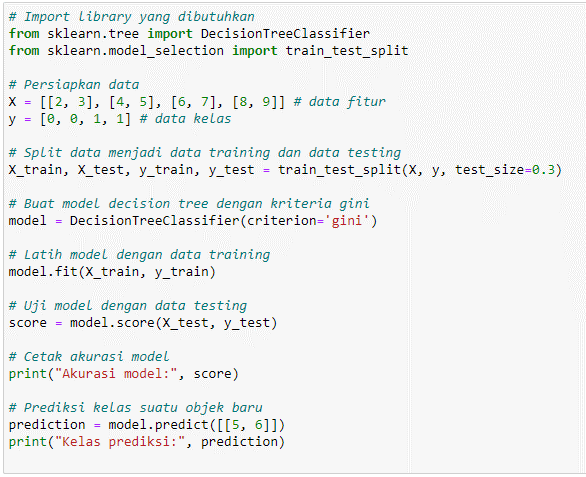
1. *Import library* yang dibutuhkan, yaitu ***GaussianNB*** dan ***train\_test\_split*** dari *scikit-learn.*
2. Persiapkan data fitur dan kelas yang akan digunakan untuk melatih model.
3. *Split* data menjadi data training dan data *testing* dengan ukuran data *testing* sebesar 30%.
4. Buat model *Naïve 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. **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*:



*Gambar 1.2.5 Contoh Script Decision Tree*

Penjelasan *script Decision Tree* diatas:

*Script* tersebut terdiri dari beberapa langkah:

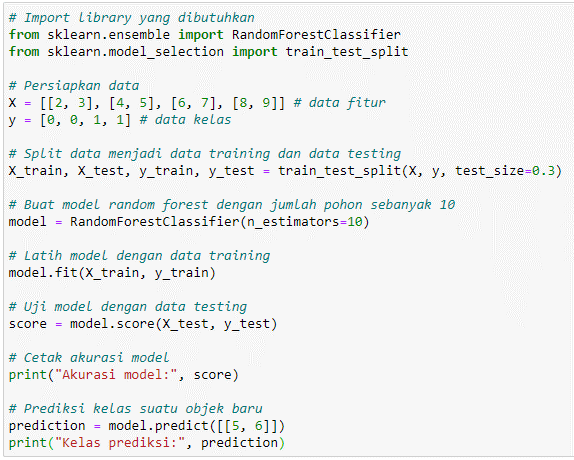
1. *Import library* yang dibutuhkan, yaitu ***DecissionTreeClassifier*** dan ***train\_test\_split*** dari *scikit-learn.*
2. Persiapkan data fitur dan kelas yang akan digunakan untuk melatih model.
3. *Split* data menjadi data training dan data *testing* dengan ukuran data *testing* sebesar 30%.
4. 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. **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 ditumbuhkan sehingga terbentuk hutan (*forest*), kemudian analisis dilakukan pada kumpulan pohon tersebut.

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



*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.

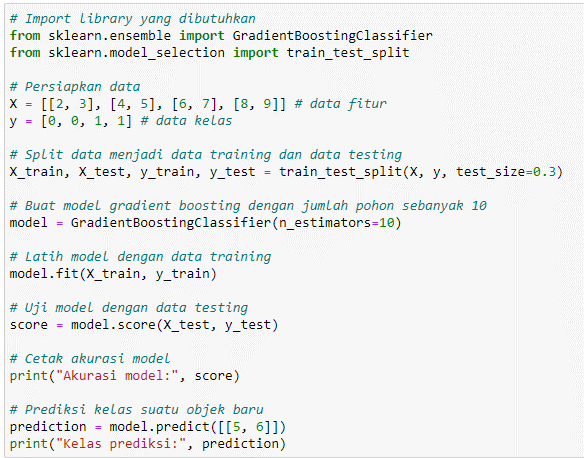
Setelah melakukan semua langkah tersebut, Anda akan mendapatkan model *Random Forest* yang telah dilatih dan diuji serta dapat digunakan untuk melakukan prediksi kelas suatu objek baru.

* + 1. **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*:



*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.

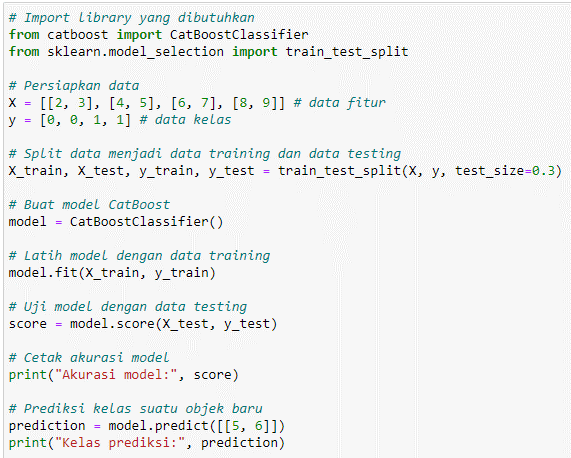
Setelah melakukan semua langkah tersebut, Anda akan mendapatkan model *Gradient Boosting* yang telah dilatih dan diuji serta dapat digunakan untuk melakukan prediksi kelas suatu objek baru.

* + 1. **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 batasan tersebut 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*:



*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. **Python**

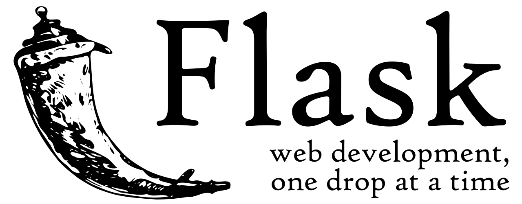
*Python* adalah bahasa yang dirancang dengan baik yang dapat digunakan secara nyata pemrograman dunia. *Python* adalah tingkat yang sangat tinggi, dinamis, berorientasi objek, bahasa pemrograman tujuan umum yang menggunakan juru bahasa dan dapat digunakan dalam domain yang luas aplikasi. *Python* dirancang 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 popular 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 diidentifikasikan 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]

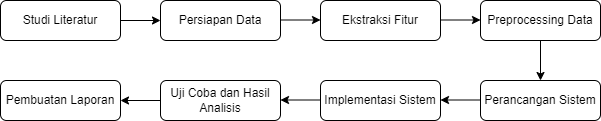
## BAB 3

## 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 ektraksi, sementara pada Sistem Evaluasi akan ada menjadi proses pengoptimalan parameter dari algoritma klasifiasi 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 ektraksi 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. Ektraksi fitur dilakukan untuk mengekstraksi fitur terdapat di situs web berdasarkan dataset yang diperoleh dari *Kaggle*. Hasil ektraksi 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 dinguakan dalam klasifikasi situs web phising. 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* sebagi 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.

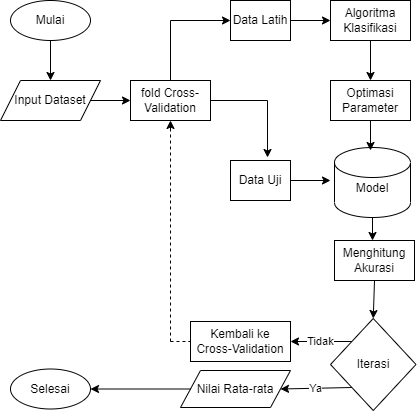
**3.2 Tahapan – tahapan Diagram Alur Metofologi Penelitian**

Pada tahapan penelitian ini dilakukan untuk mengektrak 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 akan dilakukan 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 evalusi 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.

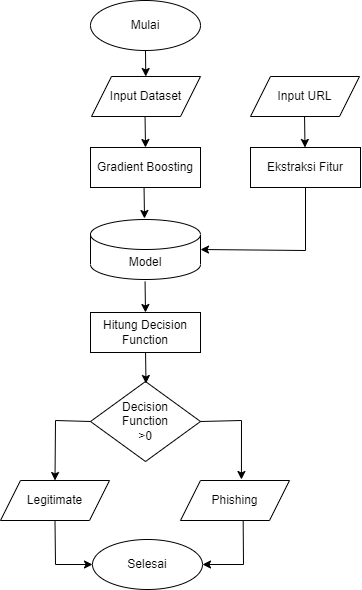
Pengujian untuk pembuatan model. Pengujian ini dilakukan dengan membandingkan kinerja model yang telah dilatih dengan dataset pengujian yang tidak digunakan sebelumnya. Hal ini dilakukan untuk mengevaluasi kinerja model dengan data yang belum pernah dilihat sebelumnya.

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 menbangun *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**

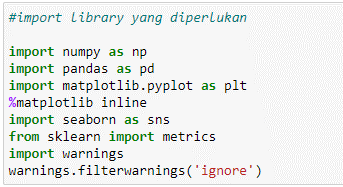
Metodologi data science untuk proyek deteksi halaman website phishing menggunakan algoritma Machine Learning Gradient Boosting Classifier adalah sebagai berikut:

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**



*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

1. *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

1. *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 peritah berikut:

Conda install –c conda-forge matplotlib

* Untuk penggunaan via pip dapat menggunakan perintah perikut:
* Python –m pip install –U pip
* Python –m pip install –U matplotlib

1. *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

1. *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.

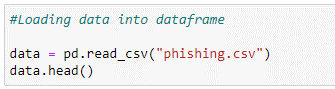
Pertama, dataset ini menyediakan 11,054 URL situs web yang merupakan contoh dari situs web yang ada di internet. Setiap sampel dalam dataset ini mewakili sebuah situs web yang unik dan memiliki URL yang sesuai.

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*.

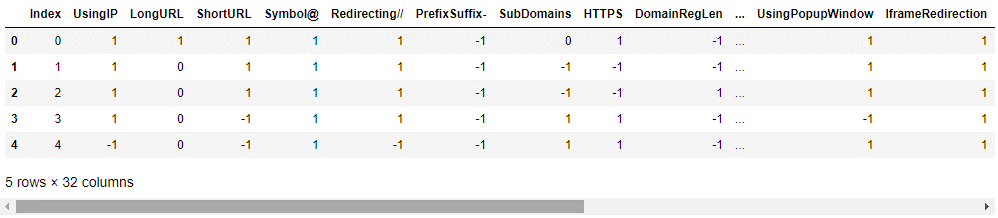
Dalam penggunaan dataset ini, para peneliti atau pengembang dapat menggunakannya untuk membuat model yang dapat mengidentifikasi situs web *phishing* dengan menganalisis parameter-parameter yang tersedia. Selain itu, dataset ini juga dapat digunakan untuk mengevaluasi performa dari model yang sudah dibuat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



*Gambar 4.2.1 Memuat Dataset*

Hasil:



*Gambar 4.2.2 Hasil Memuat Dataset*

**4.3 Features Pada Data Set**

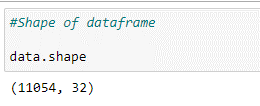
Berikut ini merupakan penjelasan semua fitur yang terdapat didalam dataset:

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*.

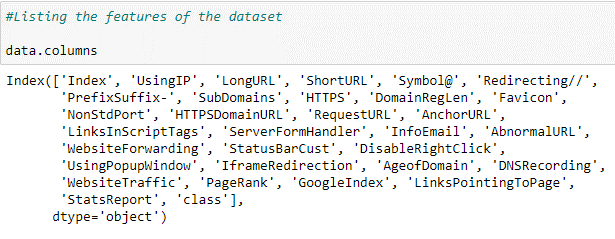
**4.4 Familiar dengan Data & EDA**

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.



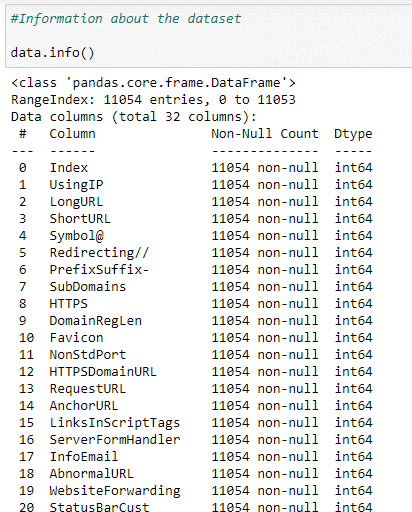
*Gambar 4.3.1 Data Shape*

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.



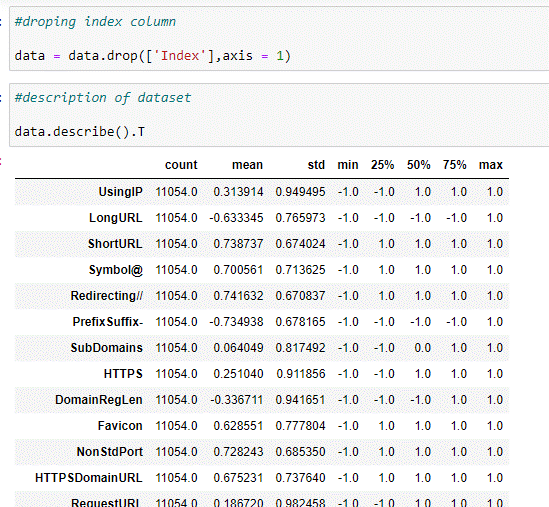
*Gambar 4.3.2 Data Columns*

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.



*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.

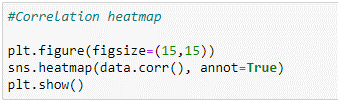


*Gambar 4.3.4 Data Info*

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.

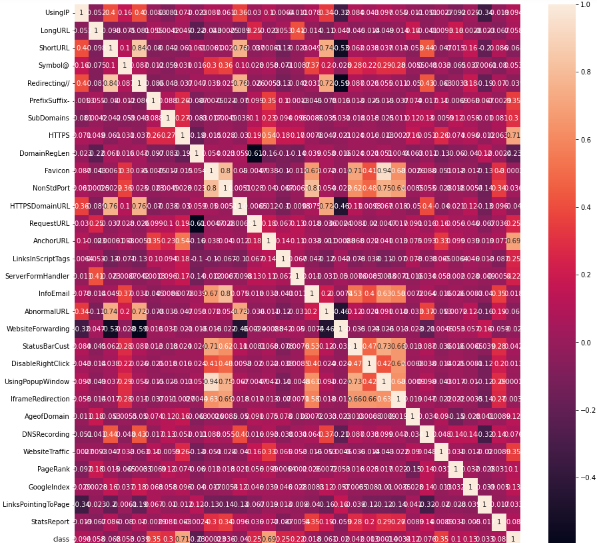
Method **T** merupakan method yang dimiliki oleh object DataFrame yang digunakan untuk men-transpose tabel, yaitu menukar posisi baris dan kolom.

**4.5 Memvisualisasikan Data**

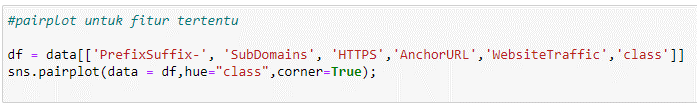


*Gambar 4.4.1 Correlation heatmap*

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().**

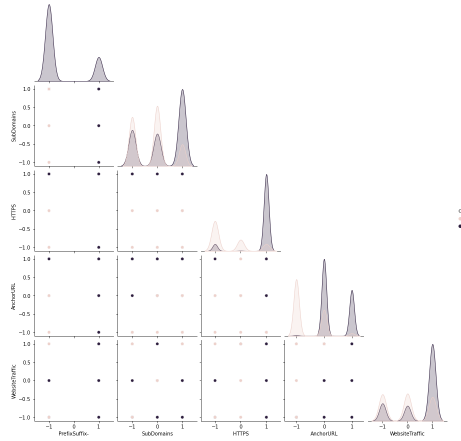


*Gambar 4.4.2 Hasil Correlation heatmap*

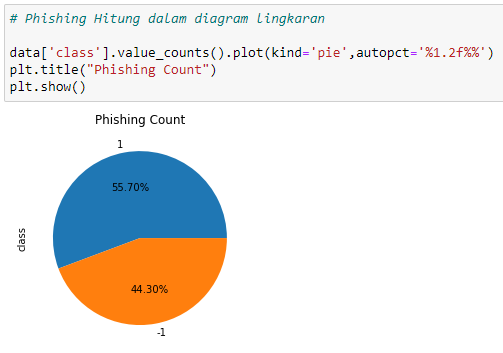


*Gambar 4.4.3 Pairplot*

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.



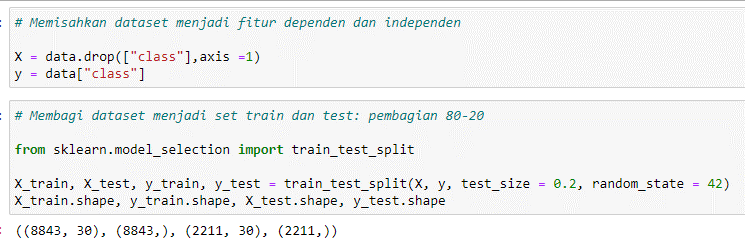
*Gambar 4.4.4 Hasil Pairplot*



*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 Mimisahkan Dataset**



*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.

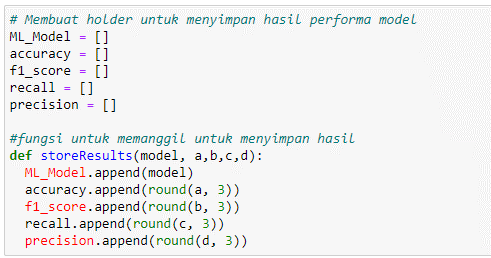
**4.7 Membangun dan Melatih Model**

*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 Clasifier*
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*.

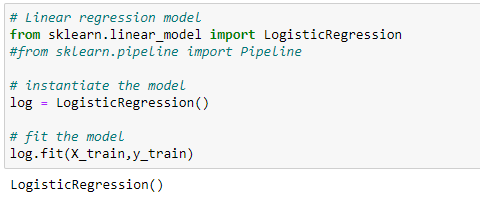


*Gambar 4.6 Membuat Holder Model*

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**



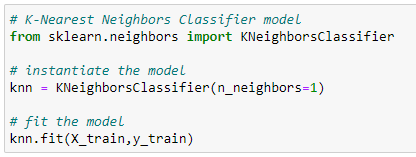
*Gambar 4.6.1 Model Logistic Regression*

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 *LogisticRegressio*n 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 membuat scriptnya.

**4.7.2 Model k-Nearest Neighbors**



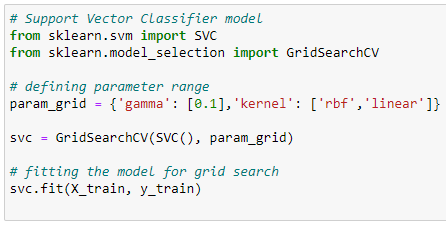
*Gambar 4.6.2 Model KNN*

*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**



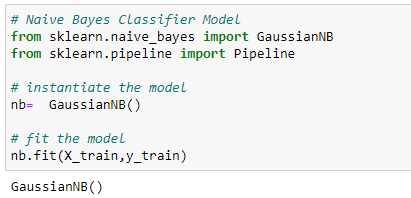
*Gambar 4.6.3 Model SVM*

*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 membuat scriptnya.

**4.7.4 Model Naïve Bayes**

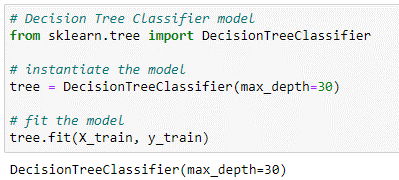


*Gambar 4.6.4 Model Naïve Bayes*

*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**

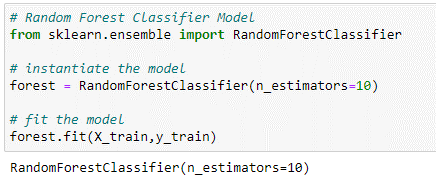


*Gambar 4.6.5 Model Decision Tree*

*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**

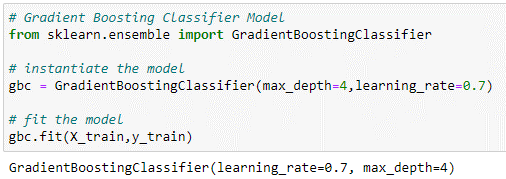


*Gambar 4.6.6 Model Random Forest*

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**

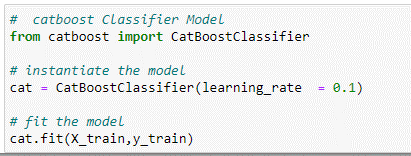


*Gambar 4.6.7 Model Gradient Boosting*

*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**

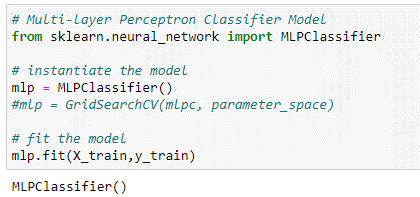


*Gambar 4.6.8 Model Catboost*

*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**



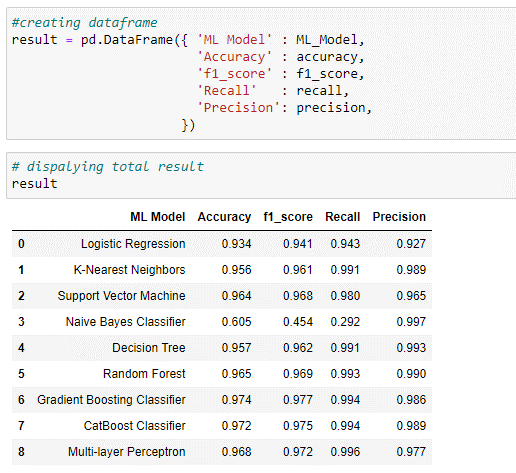
*Gambar 4.6.8 Model Multilayer Perceptrons*

*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**

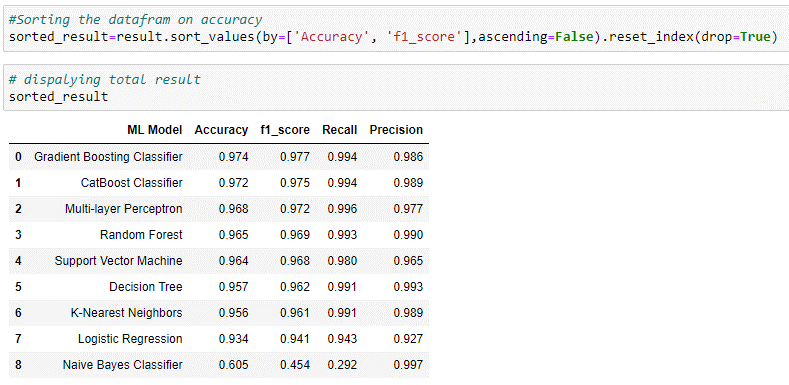


*Gambar 4.7.1 Membuat Dataframe Perbandingan Model*

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.

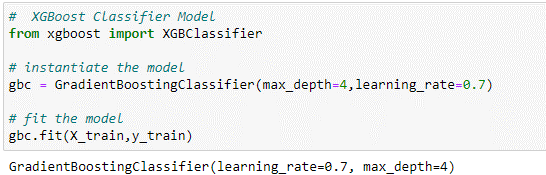


*Gambar 4.7.2 Sorting Dataframe Accuracy*

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**

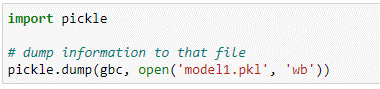


*Gambar 4.8 Menyimpan Model Terbaik*

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.

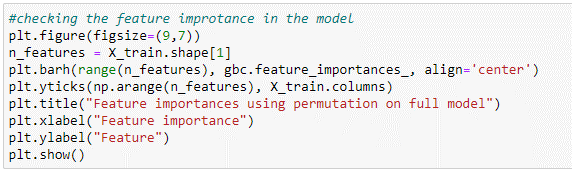
Setelah *object gbc* terbuat, dilakukan *training* model dengan menggunakan *method f****it()***. *Method* ini membutuhkan dua argumen yaitu *X\_train* dan *y\_train* yang merupakan *set train* yang telah dibuat sebelumnya. Setelah *training* selesai, maka model yang telah dilatih akan tersimpan dalam *object gbc*.

**4.10 Ubah Menjadi Format Pickle**



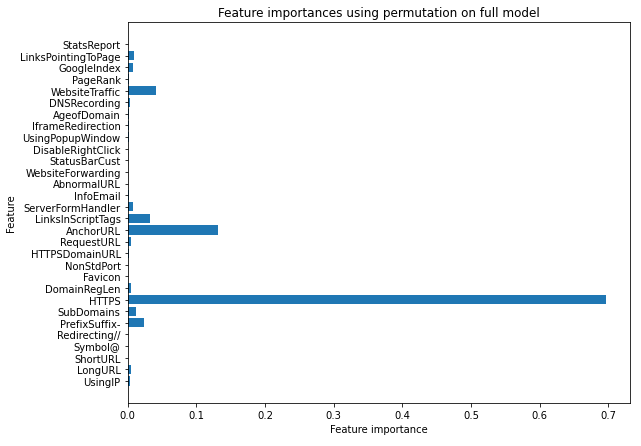
*Gambar 4.9.1 Import Library pickle*

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**).



*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**

1. **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]

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

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

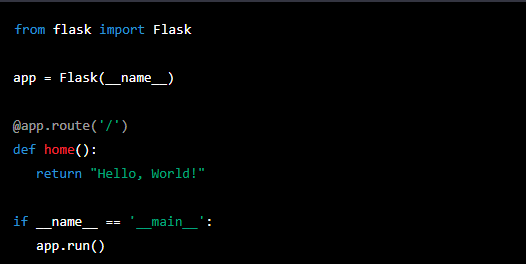
**Hasil Penelitian:** Didapat hasil *Decision Tree* dengan nilai *accuracy* 85.40%*.* [19]

# 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:

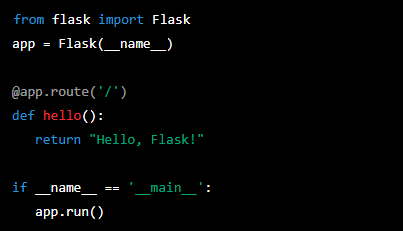


*Gambar 5.1 Script Flask*

* 1. **Install Flask**

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

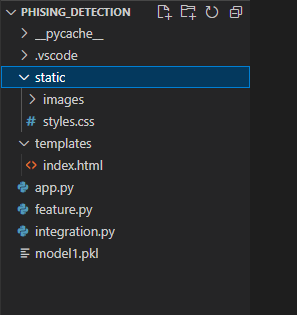
1. Buka *command prompt* atau terminal Anda.
2. Aktifkan *virtual environment* jika Anda menggunakannya. Jika belum memiliki *virtual environment*, Anda dapat membuatnya dengan perintah *"python -m venv* nama\_*virtual\_environment*".
3. *Install Flask* dengan perintah "*pip install flask*". Ini akan 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:



*Gambar 5.1.1 Cek Flask Sudah Berjalan*

1. 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/".
2. Buka browser Anda dan akses alamat tersebut. Anda akan melihat output "Hello, Flask!" pada halaman web.
   1. **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 *phising.*Berikut merupakan gambaran struktur sederhana dalam pembuat aplikasi *website* pendeteksi *phishing* ini.



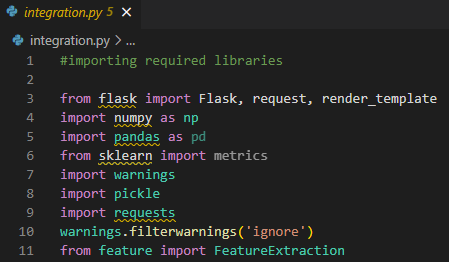
*Gambar 5.2.1 Struktur Folder Flask*

* 1. **Import Pickle**

Silahkan perhatikan kembali pada struktur folder diatas, disana terdapat **model1.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 phising bedasarkan model yang kita buat sebelumnya dan kita tentukan yaitu *Gradient Boosting Classifier.*

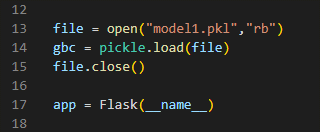
Selanjutnya kita akan bermain-main dengan file **integration.py**, dimana file ini akan digunakan untuk dapat menggunakan model yang sudah kita simpan kedalam pickle. Berikut merupakan script”nya:

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



*Gambar 5.3.1 Import Library Flask*

1. Lalu panggil *model1.pkl* dan lakukan load untuk filenya.

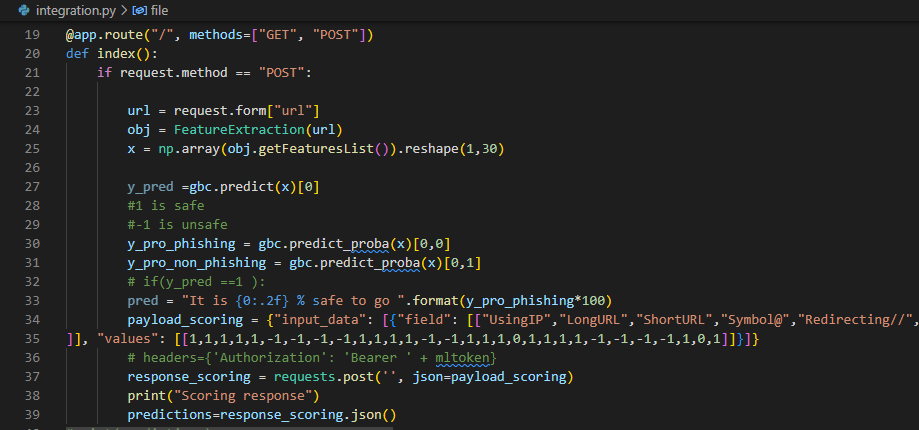


*Gambar 5.3.2 Panggil Model*

*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*.

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



*Gambar 5.3.3 Ektraksi Model*

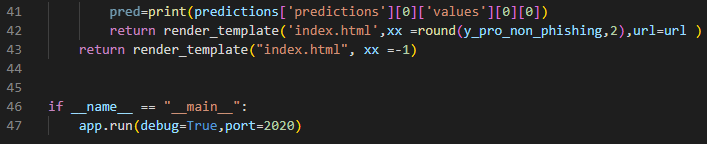
1. Kirim Hasilnya kedalam template *index.html* dengan menggunakan *render*, dan akses via web *browser*.

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.



*Gambar 5.3.4 Render Model*

* 1. **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*.

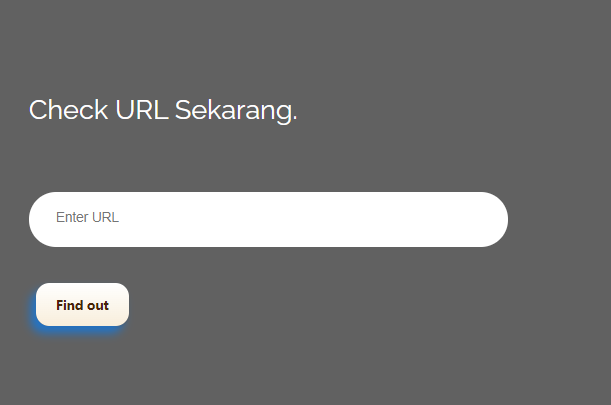
Di bagian cek URL, pengguna dapat memasukkan URL yang ingin dicek. Setelah URL dimasukkan, sistem akan menganalisis URL tersebut menggunakan model yang telah dibuat sebelumnya. Model ini akan mengevaluasi URL tersebut dan memberikan hasil yang dapat diterima oleh pengguna. Hasil dari analisis ini dapat 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*

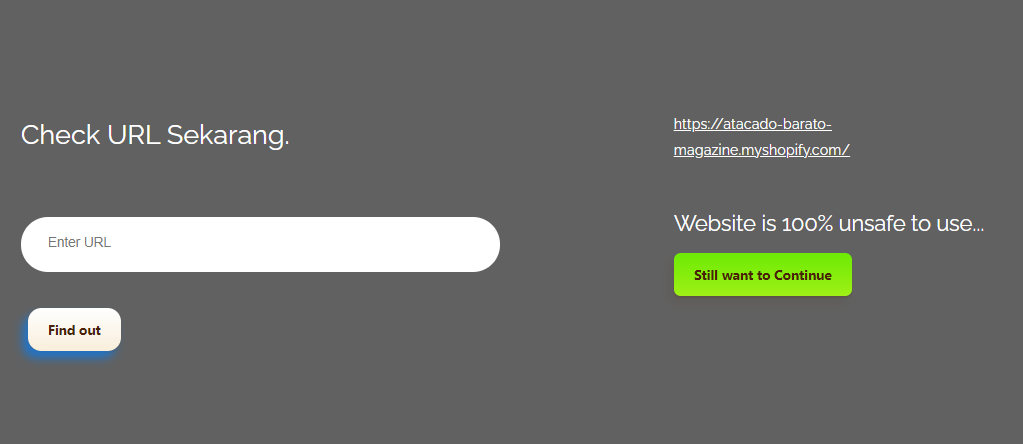
Sebagai contoh pertama, kita akan mencoba untuk memasukkan URL *https://atacado-barato-magazine. myshopify.com/* pada kolom pengecekan yang tersedia. Setelah URL dimasukkan, kita akan mengklik tombol "*find out*" untuk memulai proses analisis.

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*

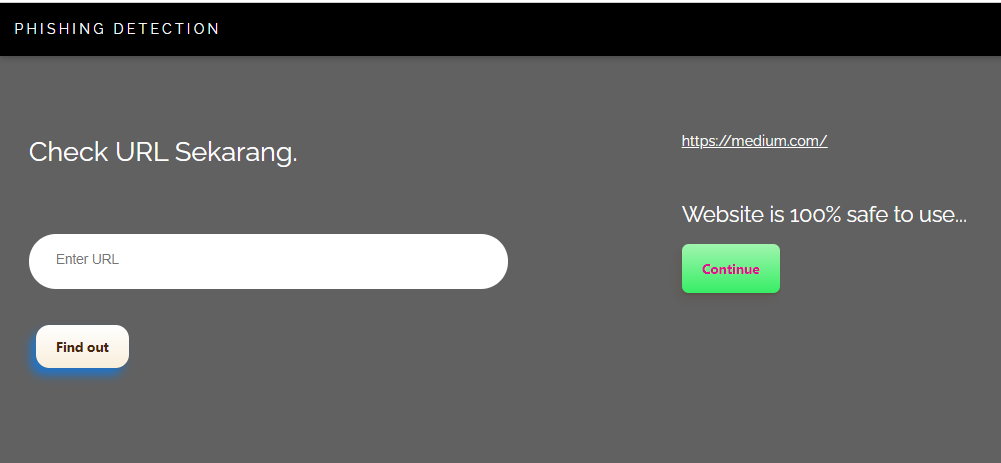
Selanjutnya, kita akan mencoba untuk melakukan pengecekan kedua dengan menggunakan contoh website dari Medium yaitu [*https://medium.com*](https://medium.com). Sama seperti pada contoh sebelumnya, kita akan memasukkan URL tersebut pada kolom pengecekan dan mengklik tombol "*find out*" untuk memulai proses 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. “*PENGGUNAAN PIRANTI LUNAK JUPYTER NOTEBOOK DALAM UPAYA MENSOSIALISASIKAN OPEN SCIENCE*”. Universitas Sriwijaya, Jl. Raya Palembang Prabumulih KM 32, Indralaya 30662.

[15] Permana Yudi, Romadlon Puji, 2019. “*PERANCANGAN SISTEM INFORMASI PENJUALAN PERUMAHAN MENGUNAKAN 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 PAGAR ALAM*”. 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-

**GLOSARIUM**

A

**Algoritma:** Suatu serangkaian instruksi untuk melakukan tugas tertentu secara logis dan sistematis. Dalam machine learning, algoritma digunakan untuk melatih model dan membuat prediksi.

**Alur Kerja**: Diagram yang mewakili proses, algoritma atau sistem dengan menggunakan simbol, kotak, dan panah untuk menghubungkannya.

B

C

**Cross Validation**: Teknik validasi model yang melibatkan pembagian data pelatihan menjadi beberapa bagian (folds) dan melatih model pada setiap bagian, dengan memvalidasi pada bagian yang berbeda setiap kali.

D

**Deployment** : Proses mempublikasikan atau mengimplementasikan model machine learning ke dalam sistem atau aplikasi produksi.

E

F

**Flask**: Sebuah micro web framework di Python yang digunakan untuk membangun aplikasi web.

**Fold**: Subset dari data pelatihan yang digunakan dalam cross-validation

G

H

**Hyperplane**: Batas keputusan dalam ruang dimensi tinggi yang memisahkan kelas yang berbeda dalam masalah klasifikasi.

I

**Interval**: Rentang nilai antara dua titik tertentu.

J

K

**Kaggle:** Sebuah platform online untuk kompetisi data science dan machine learning.

 L

M

**Machine Learning**: Bidang AI yang berfokus pada pengembangan algoritma dan model yang dapat belajar dari dan membuat prediksi pada data.

**Mengkonversi Output**: Proses transformasi output dari model machine learning ke dalam bentuk yang dapat digunakan oleh sistem atau aplikasi lain.

**Microframework:** Sebuah kerangka aplikasi web kecil yang hanya menyediakan hal yang esensial untuk membangun aplikasi web.

**Multikolinearitas**: Sifat statistik dimana dua atau lebih variabel independen sangat berhubungan, sehingga sulit untuk menentukan efek individual mereka pada variabel dependen.

N

O

P

**Pelatihan** (train): Proses memberikan data ke model machine learning untuk menyesuaikan parameter dan meningkatkan akurasi.

**Pesan Instan:** Komunikasi berbasis teks real-time antara dua atau lebih pengguna melalui jaringan.

**Phishing:** Jenis serangan cyber dimana pelaku berpura-pura sebagai entitas terpercaya untuk menipu korban agar mengungkapkan informasi sensitif, seperti kata sandi atau nomor kartu kredit.

**Pickle:** Sebuah modul di Python untuk serialisasi dan deserialisasi objek Python, digunakan untuk menyimpan dan memuat model dalam machine learning.

**Precision**: Metrik evaluasi model yang mengukur seberapa baik model memprediksi item positif benar.

Q

R

**Rasio**: Perbandingan numerik dari dua kuantitas.

**Recall**: Metrik evaluasi model yang mengukur seberapa baik model menemukan semua item positif.

S

T

U

**User-friendly:** Mudah digunakan, dengan antarmuka sederhana dan intuitif bagi pengguna.

V

W

**Website:** Himpunan halaman web terkait yang dihosting di server dan dapat diakses melalui internet.

X

Y

**INDEKS**

**A**

Algoritma 3, 4, 8

Alur Keja 27

**B**

**C**

Cross Validation 29

**D**

Deployment 33

**E**

**F**

Flask 67, 68, 69, 70, 71

**G**

**H**

Hyperplane 8

**I**

Interval 4

**J**

**K**

Kaggle 27, 37

**L**

**M**

Machine Learning 1, 2, 3, 21

Microframework 21

Multikolinearitas 5, 16

**N**

**O**

**P**

Phishing 1, 14, 21, 29

Pickle 24, 63, 69

**Q**

**R**

Rasio 4

**S**

**T**

**U**

User-friendly 20

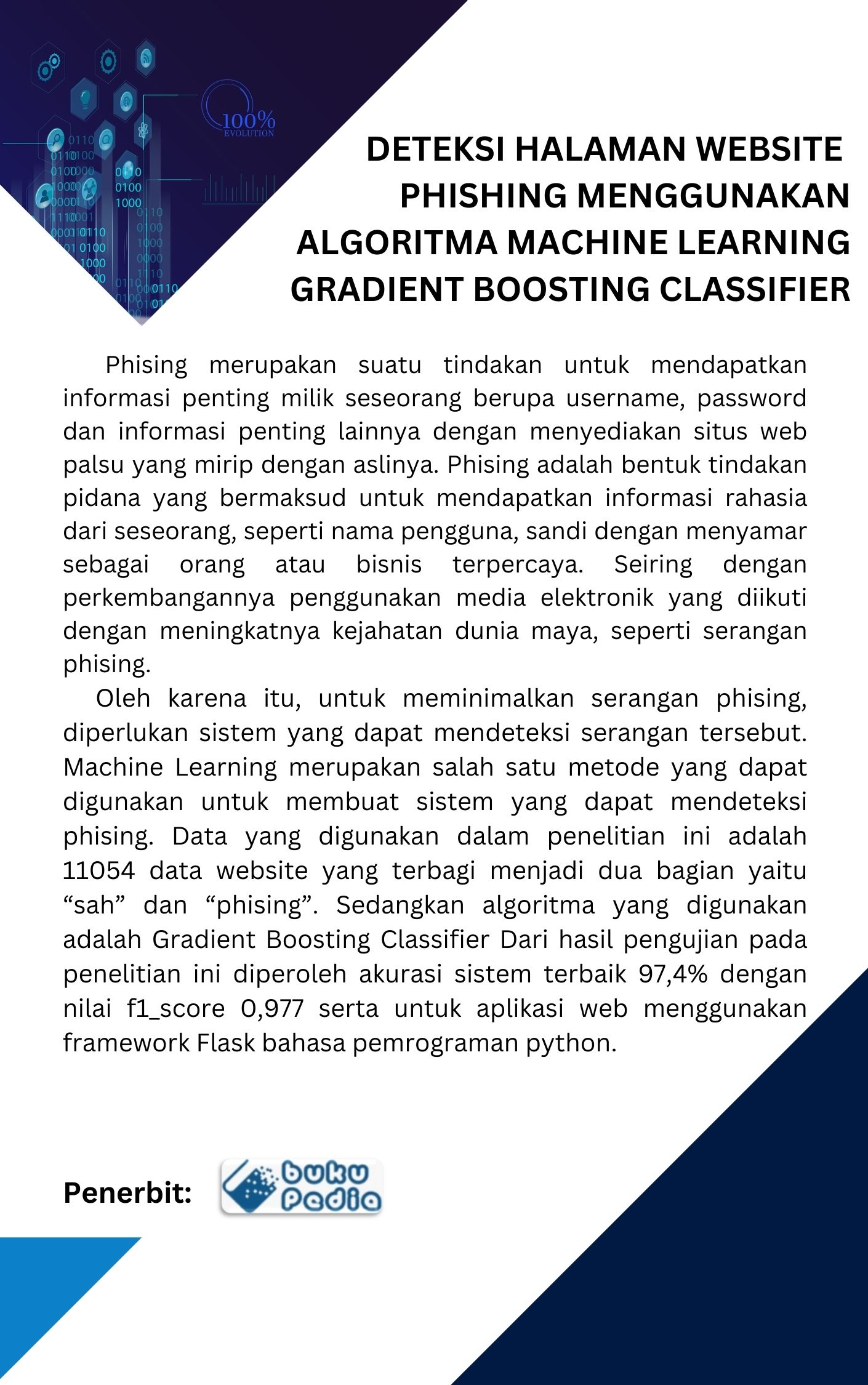
**TENTANG PENULIS**

Muhammad Rizal Supriadi, seorang penulis yang berasal dari Bandung, lahir pada tanggal 17 September 2000. Dari kecil, Saya memiliki minat dan bakat yang luar biasa dalam dunia pendidikan. Pendidikan formalnya dimulai dari tingkat dasar hingga saat ini ditempuh di Bandung, menunjukkan dedikasi dan tekad yang tinggi untuk terus belajar dan meningkatkan ilmu pengetahuannya.

Setelah menyelesaikan pendidikan di sekolah menengah kejuruan, Saya memilih untuk melanjutkan studinya di ULBI (Universitas Logistik dan Bisnis Internasional) dengan memilih jurusan D4 Teknik Informatika. Dalam dunia akademis, Saya dikenal sebagai siswa yang selalu ingin belajar dan memiliki pengetahuan yang luas.

Selain menekuni dunia pendidikan, Saya juga memiliki hobi yang sangat menyenangkan dan berguna bagi kesehatannya. Saya sangat suka olahraga dan memiliki gaya hidup yang sehat. Dalam bidang bisnis, Saya juga memiliki minat dan bakat yang sangat baik serta memiliki visi dan misi untuk membangun bisnis yang sukses dan memberikan manfaat bagi masyarakat sekitar.

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

****