

# 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:

```
from flask import Flask

app = Flask(__name__)

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

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

*Gambar 5.1 Script Flask*

## 5.1 Install Flask

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

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

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

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

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

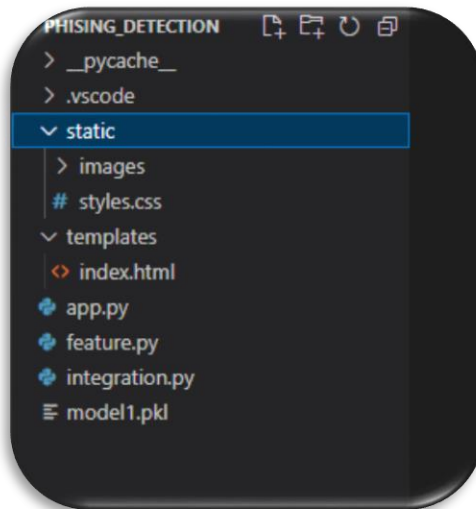
Gambar 5.1.1 Cek Flask Sudah Berjalan

5. Jalankan file tersebut dengan perintah "*python nama\_file.py*". Jika *Flask* sudah terinstall dengan benar, Anda akan mendapatkan output "*Running on http://127.0.0.1:5000/*".
6. Buka browser Anda dan akses alamat tersebut. Anda akan

melihat output "Hello, Flask!" pada halaman web.

## 5.2 Membuat Struktur Folder

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



Gambar 5.2.1 Struktur Folder Flask

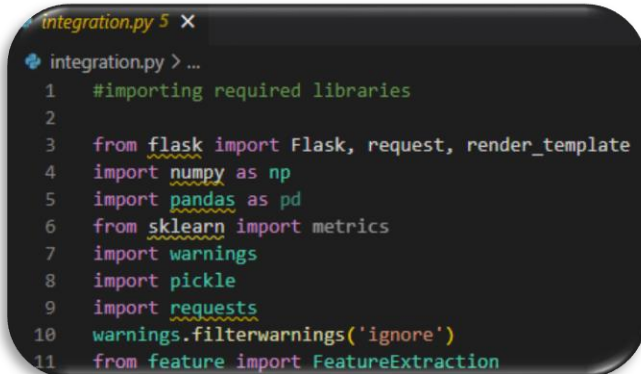
## 5.3 Import Pickle

Silahkan perhatikan kembali pada struktur folder diatas, disana terdapat **model1.pkl**, Dimana model tersebut merupakan hasil *convert* model yang kita buat sebelumnya di *jupyter notebook*. Kita akan menggunakan atau mengonsumsi model tersebut untuk dapat menentukan web phishing berdasarkan model yang kita buat sebelumnya dan kita tentukan yaitu *Gradient Boosting Classifier*.

Selanjutnya kita akan bermain-main dengan file **integration.py**, dimana file ini akan digunakan untuk dapat

menggunakan model yang sudah kita simpan kedalam pickle. Berikut merupakan script”nya:

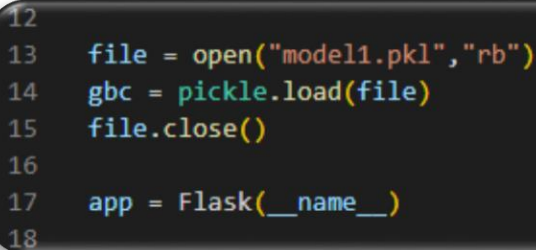
1. *Import* terlebih dahulu untuk *library* yang digunakan untuk keperluan penggunaan model.



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

Gambar 5.3.1 Import Library Flask

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



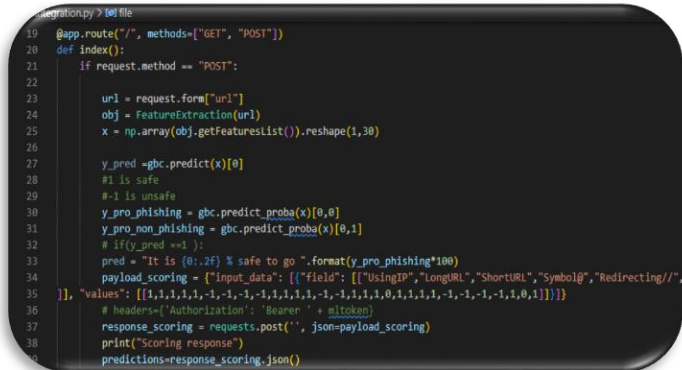
```
12
13 file = open("model1.pkl", "rb")
14 gbc = pickle.load(file)
15 file.close()
16
17 app = Flask(__name__)
18
```

Gambar 5.3.2 Panggil Model

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

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

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



```

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

```

*Gambar 5.3.3 Ekstraksi Model*

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

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

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

pengguna akan diubah menjadi tampilan yang mudah dipahami.

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

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

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

*Gambar 5.3.4 Render Model*

## 5.4 Hasil

Setelah selesai melakukan proses pengolahan data dan

pembuatan model, selanjutnya adalah melakukan pengecekan URL pada sistem deteksi *phishing*. Langkah pertama yang harus dilakukan adalah menjalankan aplikasi *Flask*.

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

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

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

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

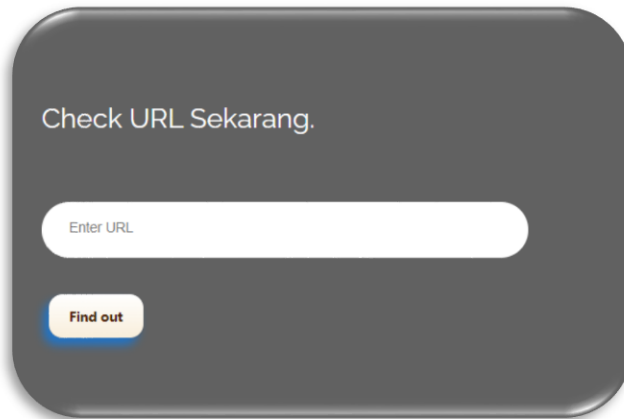


*Gambar 5.4.1 Halaman Web Awal*

Setelah diarahkan ke bagian cek URL, pengguna dapat melakukan pengecekan terhadap suatu URL yang ingin dianalisis. Hal pertama yang harus dilakukan adalah menyisipkan atau mengetikkan URL yang akan dianalisis pada kolom yang tersedia. Pengguna dapat menyisipkan URL dengan menggunakan cara *copy paste* atau dengan mengetikkan URL secara manual.

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





*Gambar 5.4.2 Cek URL*

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

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

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

Pengguna harus mengambil tindakan yang tepat untuk menghindari kerugian yang mungkin terjadi. Tindakan yang

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

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



*Gambar 5.4.3 Hasil URL Phishing*

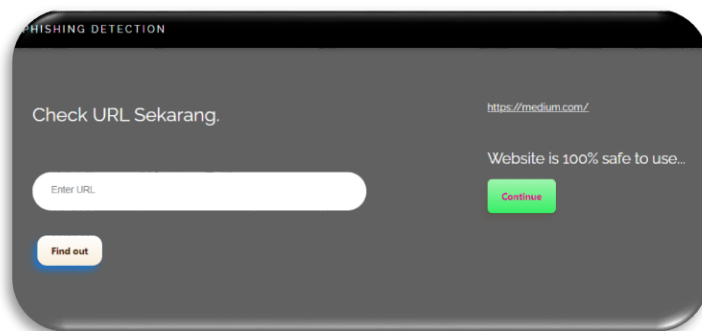
Selanjutnya, kita akan mencoba untuk melakukan pengecekan kedua dengan menggunakan contoh website dari Medium yaitu <https://medium.com>. Sama seperti pada contoh sebelumnya, kita akan memasukkan URL tersebut pada kolom pengecekan dan mengklik tombol "*find out*" untuk memulai proses analisis.

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

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

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

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



*Gambar 5.4.4 Hasil URL tidak Phishing*

## 5.5 Kesimpulan

Secara keseluruhan, sistem deteksi *phishing* adalah alat

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

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