BAB 4
Pembuatan Model Machine Learning

4.1 Import Library Tahap Awal

```
#import library yang diperlukan

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

Gambar 4.1 Import Library Tahap Awal

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

1. *NumPy*: *Library* ini menyediakan fungsi-fungsi matematika yang berguna untuk melakukan operasi pada *array*.
   Cara *install*: untuk menginstal *library numpy* anda dapat menginstallnya dengan menggunakan ***Conda*** dan ***Pip*****.**
   - Ketika menggunakan conda langkah pertama jalankan perintah-perintah berikut:

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

```
- conda create -n my-env
```

```
- conda activate my-env

- # If you want to install from conda-forge

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

- # The actual install command

- conda install numpy
```

- Ketika menggunakan Pip dapat menjalankan perintah berikut:
  `pip install numpy`

2. *pandas*: *Library* ini memungkinkan kita untuk membaca dan mengelola data serta menyediakan berbagai fungsi untuk mengeksplorasi data.
   Cara install:
   - Untuk menginstall pandas kita bisa menggunakan perintah berikut:
     `pip install pandas`
3. *Matplotlib*: Library ini memungkinkan kita untuk membuat berbagai jenis plot, seperti plot garis, plot bar, dan lain-lain.
   Cara install: Untuk menginstall *matplotlib* kita dapat

menggunakan 2 cara yaitu dengan menggunakan **Conda** dan **Pip**.

- Untuk pengguna conda dapat menjalankan perintah berikut:

  Conda install matplotlib

  Untuk penggunaan conda forge dapat menggunakan 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

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

   Cara install: Untuk menginstall seaborn kita dapat menggunakan ***Conda*** dan **Pip.**
   - Untuk penggunaan Conda dapat menjalankan perintah berikut:
     - Conda install seaborn
     - Conda install seaborn –c conda-forge
   - Untuk penggunaan via Pip dapat menjalankan perintah berikut:
     - Pip install seaborn
5. *sklearn*: *Library* ini merupakan library yang menyediakan berbagai algoritma *machine learning* yang berguna untuk melakukan pembelajaran mesin.

   Cara *install*: Untuk cara penginstallan sklearn dapat menggunakan perintah berikut:
   - pip install –U scikit-learn

Pada akhir *script*, terdapat perintah

"*warnings.filterwarnings('ignore')*" yang berfungsi untuk menonaktifkan *warning* yang mungkin muncul saat menjalankan program.

4.2 Memuat Dataset

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

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

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

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

Kedua, setiap sampel dalam dataset ini juga memiliki label kelas yang mengidentifikasinya sebagai situs web *phishing* atau bukan. Label kelas ini dapat berupa 1 atau -1, dimana 1 menandakan bahwa situs web tersebut adalah situs web *phishing*, sedangkan -1 menandakan bahwa situs web tersebut bukan situs web *phishing*.

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

Dalam penggunaan dataset ini, para peneliti atau

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

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

```
#Loading data into dataframe

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

Gambar 4.2.1 Memuat Dataset

Hasil:

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

5 rows × 32 columns

Gambar 4.2.2 Hasil Memuat Dataset

4.3 Features Pada Data Set

Berikut ini merupakan penjelasan semua fitur yang terdapat didalam dataset:

1. ***Index***: Menunjukkan apakah halaman website terindex oleh mesin pencari seperti Google atau tidak. Halaman *phishing* mungkin tidak terindex oleh mesin pencari.
2. ***UsingIP***: Menunjukkan apakah halaman *website*

menggunakan alamat IP atau nama domain. Halaman *phishing* mungkin menggunakan alamat IP.

3. ***LongURL***: Menunjukkan panjang URL dari halaman *website*. Halaman *phishing* mungkin memiliki URL yang panjang dan rumit.
4. ***ShortURL***: Menunjukkan apakah halaman *website* menggunakan layanan URL pendek seperti bit.ly. Halaman phishing mungkin menggunakan layanan URL pendek untuk menyembunyikan alamat sebenarnya.
5. ***Symbol@***: Menunjukkan apakah halaman *website* menggunakan simbol @ dalam URL. Halaman phishing mungkin menggunakan simbol @ dalam URL untuk menyembunyikan alamat sebenarnya.
6. ***Redirecting//***: Menunjukkan apakah halaman *website* mengarahkan ke URL lain. Halaman phishing mungkin mengarahkan ke URL lain untuk menipu pengguna.
7. ***PrefixSuffix-***: Menunjukkan apakah halaman *website* menggunakan prefiks atau sufiks dalam URL. Halaman *phishing* mungkin menggunakan *prefiks* atau *sufiks* dalam URL untuk menyembunyikan alamat sebenarnya.
8. ***SubDomains***: Menunjukkan apakah halaman *website* menggunakan subdomain. Halaman *phishing* mungkin menggunakan subdomain untuk menipu pengguna.
9. **HTTPS**: Menunjukkan apakah halaman *website* menggunakan protokol HTTPS atau tidak. Halaman *phishing* mungkin tidak menggunakan HTTPS.
10. ***DomainRegLen***: Menunjukkan berapa lama domain *website* terdaftar. Halaman *phishing* mungkin memiliki domain yang baru terdaftar.
11. ***Favicon***: Menunjukkan apakah halaman *website* memiliki *favicon* atau tidak. Halaman *phishing* mungkin tidak

memiliki *favicon*.

12. ***NonStdPort***: Menunjukkan apakah halaman *website* menggunakan *port non-standar*. Halaman *phishing* mungkin menggunakan port *non-standar* untuk menyembunyikan alamat sebenarnya.
13. ***HTTPSDomainURL***: Menunjukkan apakah alamat *website* menggunakan protokol HTTPS atau tidak. Halaman *phishing* mungkin tidak menggunakan HTTPS.
14. ***RequestURL***: Menunjukkan apakah halaman *website* mengirim permintaan ke server. Halaman *phishing* mungkin mengirim permintaan yang tidak diinginkan ke server.
15. ***AnchorURL***: Menunjukkan apakah halaman *website* mengandung *link-link* yang dapat mengarahkan pengguna ke halaman lain. Halaman *phishing* mungkin mengandung *link-link* yang dapat mengarahkan pengguna ke halaman *phishing* lainnya.
16. ***LinksInScriptTags***: Menunjukkan apakah halaman *website* mengandung *link* yang tersembunyi di dalam *tag script*. Halaman *phishing* mungkin mengandung *link* yang tersembunyi di dalam tag *script* untuk menipu pengguna.
17. ***ServerFormHandler***: Menunjukkan apakah halaman *website* mengandung *form* yang ditangani oleh server. Halaman *phishing* mungkin mengandung form yang ditangani oleh *server* untuk mengumpulkan informasi pengguna.
18. ***InfoEmail***: Menunjukkan apakah halaman *website* menyediakan informasi kontak *email*. Halaman *phishing* mungkin tidak menyediakan informasi kontak *email*.
19. ***AbnormalURL***: Menunjukkan apakah halaman *website* memiliki URL yang tidak biasa. Halaman *phishing*

mungkin memiliki URL yang tidak biasa untuk menyembunyikan alamat sebenarnya.

20. ***WebsiteForwarding***: Menunjukkan apakah halaman *website* mengarahkan pengguna ke halaman lain. Halaman *phishing* mungkin mengarahkan pengguna ke halaman *phishing* lainnya.
21. ***StatusBarCust***: Menunjukkan apakah halaman website mengubah tampilan status bar pada browser. Halaman *phishing* mungkin mengubah tampilan *status* bar untuk menipu pengguna.
22. ***DisableRightClick***: Menunjukkan apakah halaman *website* menonaktifkan klik kanan pada *mouse*. Halaman *phishing* mungkin menonaktifkan *klik* kanan untuk mencegah pengguna mengecek alamat sebenarnya.
23. ***UsingPopupWindow***: Menunjukkan apakah halaman *website* menggunakan jendela *popup*. Halaman *phishing* mungkin menggunakan jendela *popup* untuk menipu pengguna.
24. ***IframeRedirection***: Menunjukkan apakah halaman *website* mengarahkan pengguna ke halaman lain melalui *iframe*. Halaman *phishing* mungkin mengarahkan pengguna ke halaman *phishing* melalui *iframe*.
25. ***AgeofDomain***: Menunjukkan berapa lama domain *website* didaftarkan. Halaman *phishing* mungkin memiliki domain yang baru didaftarkan.
26. ***DNSRecording***: Menunjukkan apakah *domain website* memiliki catatan DNS. Halaman *phishing* mungkin tidak memiliki catatan DNS.
27. ***WebsiteTraffic***: Menunjukkan jumlah *traffic* yang datang ke *website*. Halaman phishing mungkin memiliki *traffic* yang rendah.

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

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

4.4 Familiar dengan Data & EDA

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

```
#Shape of dataframe

data.shape
```

```
(11054, 32)
```

Gambar 4.3.1 Data Shape

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

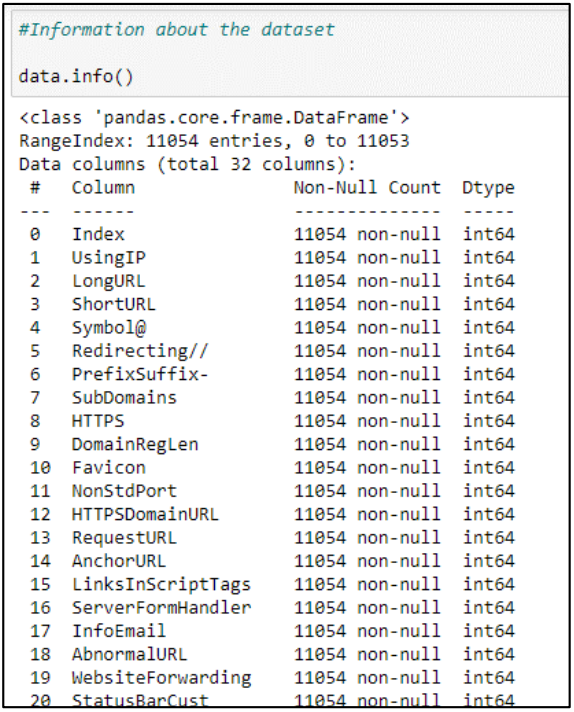
```
#Listing the features of the dataset

data.columns
```

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

Gambar 4.3.2 Data Columns

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



```
#Information about the dataset

data.info()
```

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

Gambar 4.3.3 Data Info

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

```
#droping index column

data = data.drop(['Index'],axis = 1)
```

```
#description of dataset

data.describe().T
```

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

Gambar 4.3.4 Data Info

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

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

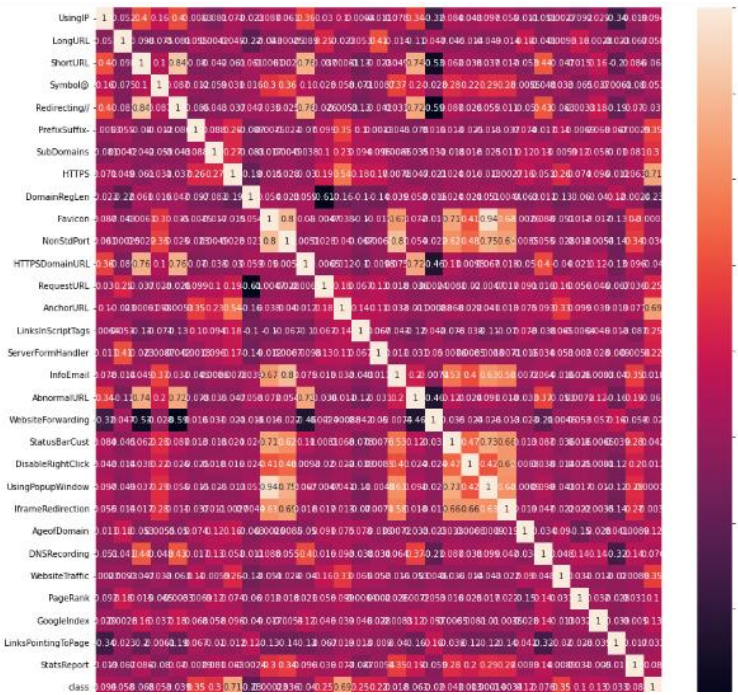
4.5 Memvisualisasikan Data

```
#Correlation heatmap

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

Gambar 4.4.1 Correlation heatmap

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



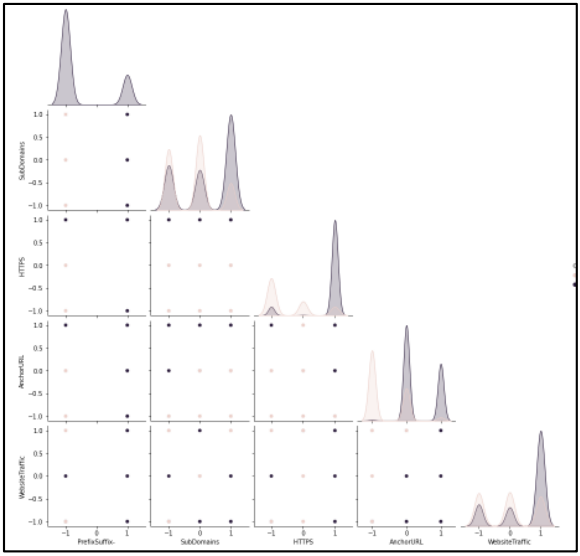
Gambar 4.4.2 Hasil Correlation heatmap

```
#pairplot untuk fitur tertentu

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

Gambar 4.4.3 Pairplot

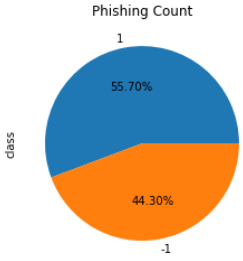
Pada script di atas, pertama-tama dilakukan pemilihan fitur yang akan dianalisis dengan menggunakan notasi *indexing* pada *object* ***DataFrame***. Kemudian, dilakukan plot pairplot dengan menggunakan perintah ***sns.pairplot().*** Argumen yang diberikan kepada perintah ini adalah data yang akan dianalisis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



Gambar 4.4.4 Hasil Pairplot

```
# Phishing Hitung dalam diagram lingkaran

data['class'].value_counts().plot(kind='pie',autopct='%1.2f%%')
plt.title("Phishing Count")
plt.show()
```



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

```
# Memisahkan dataset menjadi fitur dependen dan independen

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

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

from sklearn.model_selection import train_test_split

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

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

Gambar 4.4.5 Memisahkan Dataset

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

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

Pembagian dataset menjadi *set train* dan *set test* merupakan langkah yang penting dalam proses pembuatan model *machine learning*. *Set train* digunakan untuk melatih

model sedangkan *set test* digunakan untuk mengevaluasi model yang telah dilatih. Dengan membagi dataset menjadi *set train* dan *set test*, kita dapat memastikan bahwa model yang dihasilkan tidak hanya dapat menangani data yang telah diketahui tapi juga dapat menangani data baru yang belum pernah dilihat sebelumnya.

4.7 Membangun dan Melatih Model

Supervised adalah salah satu jenis pembelajaran mesin yang paling umum digunakan dan berhasil. Pembelajaran yang *supervised* digunakan kapan pun ingin memprediksi hasil/label tertentu dari sekumpulan fitur tertentu, dan memiliki contoh pasangan fitur-label.Membangun model pembelajaran mesin dari pasangan fitur-label ini, yang terdiri dari set train. Tujuannya adalah membuat prediksi akurat untuk data baru.

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

1. *Logistic Regression*
2. *k-Nearest Neighbors*
3. *Support Vector 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*.

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

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

Gambar 4.6 Membuat Holder Model

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

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

4.7.1 Model Logistic Regression

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

# instantiate the model
log = LogisticRegression()

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

```
LogisticRegression()
```

Gambar 4.6.1 Model Logistic Regression

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

1. Import kelas ***LogisticRegression*** dari modul ***sklearn.linear_model***. Kelas *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

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

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

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

Gambar 4.6.2 Model KNN

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

1. Import kelas ***KNeighborsClassifier*** dari modul ***sklearn.neighbors***. Kelas ***KNeighborsClassifier*** merupakan kelas yang digunakan untuk membuat model KNN.
2. Instansiasi objek model dengan menggunakan kelas ***KNeighborsClassifier***. Objek model ini akan dibuat dengan menggunakan parameter ***n_neighbors=1***, yang menandakan bahwa model KNN yang akan dibuat akan menggunakan 1 tetangga terdekat dari setiap titik data.
3. Melatih model dengan menggunakan data latih dengan memanggil fungsi ***fit*** pada objek model yang telah dibuat. Fungsi ***fit*** akan melatih model dengan menggunakan data latih yang diberikan.

Setelah model dilatih, kita dapat menggunakannya untuk memprediksi target pada data uji dengan memanggil fungsi predict pada objek model yang telah dilatih. Selanjutnya, kita dapat mengevaluasi performa model

dengan menghitung beberapa metrik seperti akurasi, *f1-score, recall, precision* dan mulai membuat scriptnya.

4.7.3 Model SVM

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

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

svc = GridSearchCV(SVC(), param_grid)

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

Gambar 4.6.3 Model SVM

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

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

Setelah model dilatih, kita dapat menggunakannya untuk memprediksi target pada data uji dengan memanggil fungsi predict pada objek model yang telah dilatih. Selanjutnya, kita dapat mengevaluasi performa model

dengan menghitung beberapa metrik seperti akurasi, *f1-score, recall, precision* dan mulai membuat scriptnya.

4.7.4 Model Naïve Bayes

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

# instantiate the model
nb=  GaussianNB()

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

```
GaussianNB()
```

Gambar 4.6.4 Model Naïve Bayes

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

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

4.7.5 Model Decision Tree

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

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

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

```
DecisionTreeClassifier(max_depth=30)
```

Gambar 4.6.5 Model Decision Tree

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

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

4.7.6 Model Random Forest

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

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

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

```
RandomForestClassifier(n_estimators=10)
```

Gambar 4.6.6 Model Random Forest

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

Setelah itu, model tersebut di-fit dengan data latih menggunakan perintah "**forest.fit(X_train, y_train)**". X_train adalah variabel yang berisi data fitur untuk data latih, sedangkan y_train adalah variabel yang berisi label untuk data latih. Dengan memanggil *method* fit(), model *Random Forest Classifier* akan dilatih dengan mempelajari korelasi antara fitur-fitur dari data dan label yang sesuai, kemudian menggunakan informasi tersebut untuk membuat sebanyak 10 pohon keputusan yang dibuat secara acak. Kemudian, hasil dari setiap pohon keputusan tersebut akan digabungkan untuk memprediksi label dari data baru.

4.7.7 Model Gradient Boosting

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

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

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

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

Gambar 4.6.7 Model Gradient Boosting

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

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

dari pohon keputusan sebelumnya.

4.7.8 Model Catboost

```
#  catboost Classifier Model
from catboost import CatBoostClassifier

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

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

Gambar 4.6.8 Model Catboost

Script di atas mengimport kelas ***CatBoostClassifier*** dari *library catboost* yang digunakan untuk membuat model *CatBoost Classifier*. *CatBoost* adalah sebuah algoritma *machine learning* yang dapat digunakan untuk melakukan klasifikasi, regresi, dan klasterisasi. Kemudian, sebuah objek dari kelas *CatBoostClassifier* dibuat dengan menggunakan perintah "**cat = *CatBoostClassifier(learning_rate=0.1)***". Perintah ini akan membuat sebuah model *CatBoost Classifier* dengan learning rate 0,1. *Learning* rate adalah seberapa cepat model belajar dari data.

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

pohon keputusan baru sesuai dengan hasil dari pohon keputusan sebelumnya.

4.7.9 Model Multilayer Perceptrons

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

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

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

```
MLPClassifier()
```

Gambar 4.6.8 Model Multilayer Perceptrons

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

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

melakukan klasifikasi.

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

4.8 Pendandingan Model

```
#creating dataframe
result = pd.DataFrame({ 'ML Model' : ML_Model,
                        'Accuracy' : accuracy,
                        'f1_score' : f1_score,
                        'Recall'   : recall,
                        'Precision': precision,
                      })
```

```
# dispalying total result
result
```

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

Gambar 4.7.1 Membuat Dataframe Perbandingan Model

Pada *script* di atas, dibuat sebuah *object result* yang merupakan sebuah ***DataFrame*** dengan menggunakan *library pandas*. ***DataFrame*** dibuat dengan menggunakan argumen yang berisi *dictionary* dengan *key* sebagai nama kolom dan *value* sebagai isi dari kolom tersebut.

DataFrame merupakan tipe data yang terdiri dari baris dan kolom yang mirip dengan tabel pada *database*. ***DataFrame*** dapat dibuat dengan menggunakan data yang berasal dari file

atau dengan memasukkan data secara manual seperti yang telah dilakukan pada ***script*** di atas.

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

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

# dispalying total result
sorted_result
```

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

Gambar 4.7.2 Sorting Dataframe Accuracy

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

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

4.9 Menyimpan Model Terbaik

```
#  XGBoost Classifier Model
from xgboost import XGBClassifier

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

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

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

Gambar 4.8 Menyimpan Model Terbaik

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

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

4.10 Ubah Menjadi Format Pickle

```
import pickle

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

Gambar 4.9.1 Import Library pickle

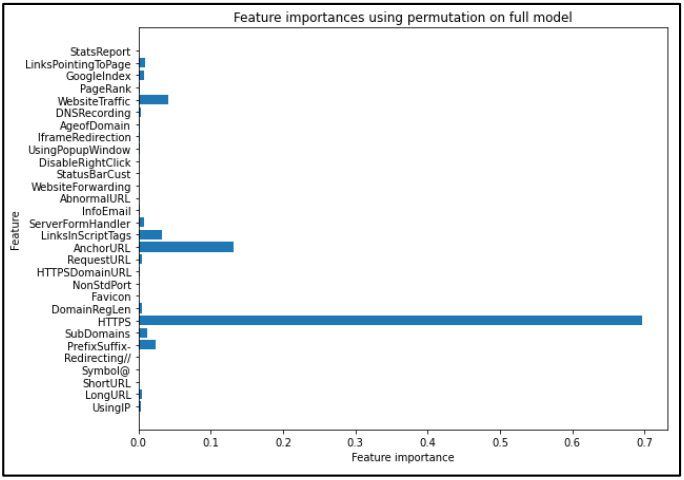
Pada script di atas, dilakukan ***import library pickle***. Kemudian, dilakukan proses serialisasi dengan menggunakan *method* ***dump()*** dari ***library pickle***. *Method* ini membutuhkan dua argumen yaitu object yang akan diserialisasi (dalam hal ini

object gbc yang merupakan model yang telah dilatih) dan file yang akan digunakan untuk menyimpan hasil serialisasi (dalam hal ini file **model1.pkl**).

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

Gambar 4.9.2 Cek Fitur pada Model

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



Gambar 4.9.3 Hasil Cek Fitur pada Model

4.11 Penelitian Sebelumnya

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

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

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

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

Judul: *Website Phising Detection Application Using*

Support Vector Machine (SVM)

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