MALWARE DETECTION MENGGUNAKAN MACHINE LEARNING
SISTEM KEAMANAN CERDAS

Oleh :

Arjun (1301204219)

PROGRAM STUDI S1 INFORMATIKA

FAKULTAS INFORMATIKA

UNIVERSITAS TELKOM

BANDUNG

2023

1. **Pendahuluan**

   Laporan ini berisi tentang penerapan algoritma naive bayes dan svm dalam deteksi malware dengan machine learning. Deteksi malware merupakan salah satu tantangan yang ada dalam dunia keamanan komputer saat ini dimana tujuan dari deteksi malware ini digunakan untuk mengenali program-program ataupun data yang ada pada komputer yang berbahaya dan dapat merusak sistem dan bahkan mencuri informasi pribadi atau yang sifatnya sensitif. Algoritma pembelajaran mesin seperti Naive Bayes dan SVM telah digunakan secara luas dalam membangun model deteksi malware yang efektif.

2. **Penjelasan Algoritma**

   a. **Naive Bayes**

      Naive bayes merupakan metode pengklasifikasian berdasarkan probabilitas sederhana dan dirancang agar dapat dipergunakan dengan asumsi antar variabel penjelas saling bebas (independen). Pada algoritma ini pembelajaran lebih ditekankan pada pengestimasian probabilitas. Keuntungan algoritma naive bayes adalah tingkat nilai error yang didapat lebih rendah ketika dataset berjumlah besar, selain itu akurasi naive bayes dan kecepatannya lebih tinggi pada saat diaplikasikan ke dalam dataset yang jumlahnya lebih besar.

   b. **SVM (Support Vector Machine)**

      Support Vector Machine atau SVM adalah algoritma pembelajaran mesin yang diawasi yang dapat digunakan untuk klasifikasi dan regresi. Cara kerja SVM didasarkan pada SRM atau Structural Risk Minimization yang dirancang untuk mengolah data menjadi Hyperplane yang mengklasifikasikan ruang input menjadi dua kelas. Teori SVM diawali dengan pengelompokan kasus-kasus linier yang dapat dipisahkan dengan hyperplane dan dibagi menurut kelasnya.

3. **Implementasi**

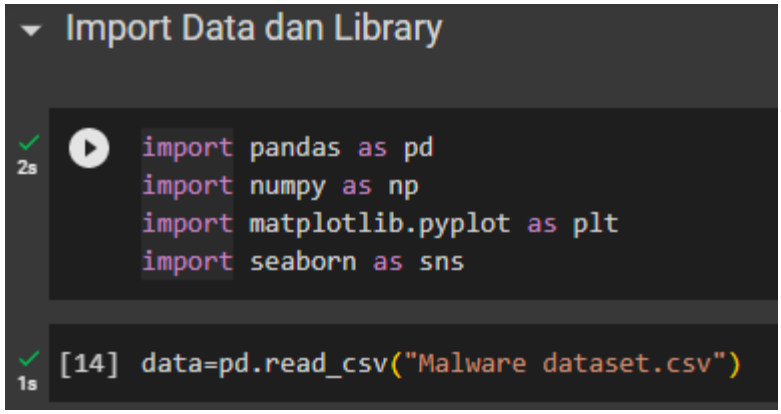
   a. **Import Library dan Dataset**

      - Library bantuan dan Mengimport dataset yaitu ‘Malware dataset.csv’ pada dataframe dengan variabel ‘data’

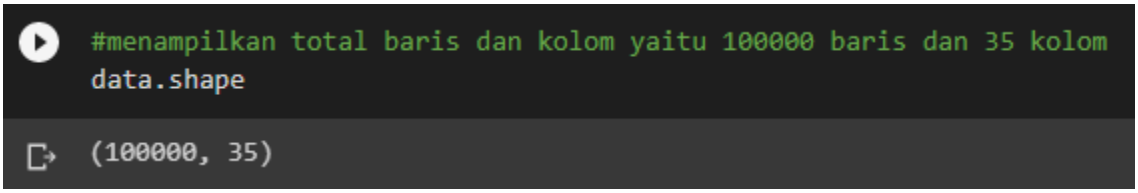
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

   b. **Exploring Data**

      - *data.head()* untuk menampilkan 5 data teratas



- *data.shape* untuk menampilkan total baris dan kolom.



- *data.isnull().sum()* untuk menunjukkan jumlah nilai null dalam setiap kolom dari dataset tersebut.

```
#menunjukkan jumlah nilai null dalam setiap kolom dari dataset tersebut
data.isnull().sum()
```

```
hash                0
millisecond         0
classification      0
state               0
usage_counter       0
prio                0
static_prio         0
normal_prio         0
policy              0
vm_pgoff            0
vm_truncate_count   0
task_size           0
cached_hole_size    0
free_area_cache     0
mm_users            0
map_count           0
hiwater_rss         0
total_vm            0
shared_vm           0
exec_vm             0
reserved_vm         0
nr_ptes             0
end_data            0
last_interval       0
nvcsw               0
nivcsw              0
min_flt             0
maj_flt             0
fs_excl_counter     0
lock                0
utime               0
stime               0
gtime               0
cgtime              0
signal_nvcsw        0
dtype: int64
```

- *data.columns* menunjukan nama-nama kolom dari dataset

```
#menunjukan nama-nama kolom dari dataset
data.columns
```

```
Index(['hash', 'millisecond', 'classification', 'state', 'usage_counter',
       'prio', 'static_prio', 'normal_prio', 'policy', 'vm_pgoff',
       'vm_truncate_count', 'task_size', 'cached_hole_size', 'free_area_cache',
       'mm_users', 'map_count', 'hiwater_rss', 'total_vm', 'shared_vm',
       'exec_vm', 'reserved_vm', 'nr_ptes', 'end_data', 'last_interval',
       'nvcsw', 'nivcsw', 'min_flt', 'maj_flt', 'fs_excl_counter', 'lock',
       'utime', 'stime', 'gtime', 'cgtime', 'signal_nvcsw'],
      dtype='object')
```

- *data1=data.dropna(how="any",axis=0) data1.head()* untuk menghapus baris axis=0 yang mengandung nilai null lalu ditampung di data1

```
#menghapus baris axis=0 yang mengandung nilai null lalu ditampugn di data1
data1=data.dropna(how="any",axis=0)
data1.head()
```

	hash	millisecond	classification	state	usage_counter	prio	s
0	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	0	malware	0	0	3069378560	
1	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	1	malware	0	0	3069378560	
2	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	2	malware	0	0	3069378560	
3	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	3	malware	0	0	3069378560	
4	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	4	malware	0	0	3069378560	

5 rows × 35 columns

c. Kalsifikasi Data Malware

- *data1["classification"].value_counts()* untuk menghitung total value dari kolom 'classification'

```
#menghitung total value dari kolom 'classification'
data1["classification"].value_counts()
```

```
malware    50000
benign     50000
Name: classification, dtype: int64
```

- *data1['classification'] = data1.classification.map({'benign':0, 'malware':1})* untuk isi dari kolom 'classification' diganti dengan numerik dimana value 'benign = 0 ' dan 'malware = 1'

```
[21] # isi dari kolom 'classification' diganti dengan numerik dimana value 'benign = 0 ' dan 'malware = 1'
     data1['classification'] = data1.classification.map({'benign':0, 'malware':1})
```

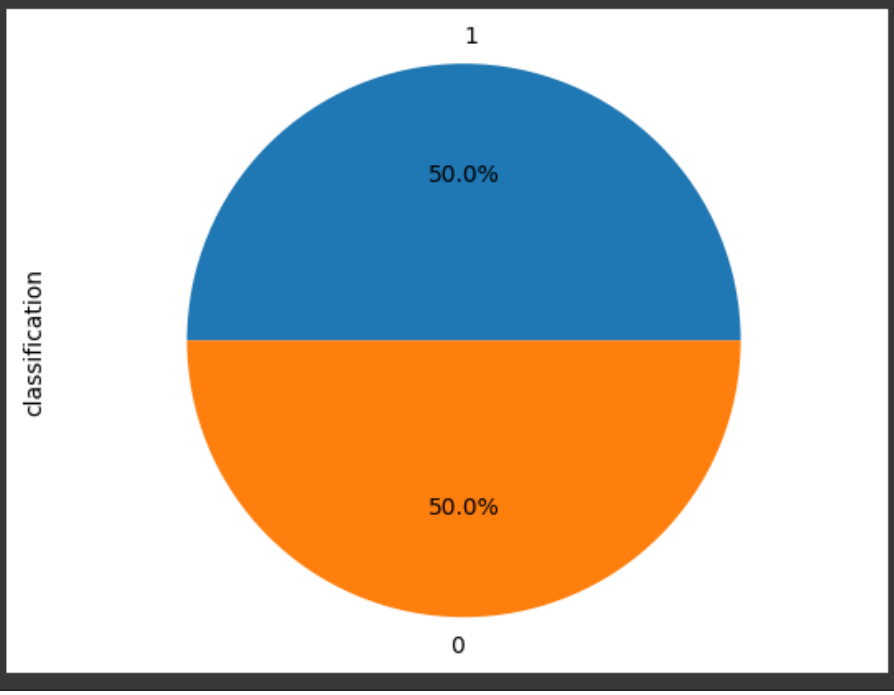
```
data1.head()
```

	hash	millisecond	classification	state	usage_counter	prio	static
0	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	0	1	0	0	3069378560	
1	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	1	1	0	0	3069378560	
2	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	2	1	0	0	3069378560	
3	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	3	1	0	0	3069378560	
4	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	4	1	0	0	3069378560	

5 rows × 35 columns

- *data1["classification"].value_counts().plot(kind="pie",autopct="%1.1f%%") plt.axis("equal") plt.show()* untuk menampilkan plot dari classification antara ‘0’ dan ‘1’

```
data1["classification"].value_counts().plot(kind="pie",autopct="%1.1f%%")
plt.axis("equal")
plt.show()
```



- *benign1=data.loc[data['classification']=='benign'] benign1["classification"].head()* untuk mengklasifikasikan data ‘benign’ dan ditampung pada variabel ‘benign1’

- *malware1=data.loc[data['classification']=='malware']*
  malware1["classification"].head() untuk mengklasifikasikan data ‘malware’ dan ditampung pada variabel ‘malware1’

```
[26] benign1=data.loc[data['classification']=='benign']
     benign1["classification"].head()

     1000    benign
     1001    benign
     1002    benign
     1003    benign
     1004    benign
     Name: classification, dtype: object

[27] malware1=data.loc[data['classification']=='malware']
     malware1["classification"].head()

     0    malware
     1    malware
     2    malware
     3    malware
     4    malware
     Name: classification, dtype: object
```

- *corr=data1.corr() corr.nlargest(35,'classification')["classification"]* untuk menghitung korelasi terhadap klasifikasi

```
corr=data1.corr()
corr.nlargest(35,'classification')["classification"]
```

```
<ipython-input-28-fabd1f11fa4f>:1: FutureWarning: The de
  corr=data1.corr()
classification       1.000000e+00
prio                 1.100359e-01
last_interval        6.952036e-03
min_flt              3.069595e-03
millisecond          5.482134e-15
gtime               -1.441608e-02
stime               -4.203713e-02
free_area_cache     -5.123678e-02
total_vm            -5.929110e-02
state               -6.470178e-02
mm_users            -9.364091e-02
reserved_vm         -1.186078e-01
fs_excl_counter     -1.378830e-01
nivcsw              -1.437912e-01
exec_vm             -2.551234e-01
map_count           -2.712274e-01
static_prio         -3.179406e-01
end_data            -3.249535e-01
maj_flt             -3.249535e-01
shared_vm           -3.249535e-01
vm_truncate_count   -3.548607e-01
utime               -3.699309e-01
nvcsw               -3.868893e-01
usage_counter                 NaN
normal_prio                   NaN
policy                        NaN
vm_pgoff                      NaN
task_size                     NaN
cached_hole_size              NaN
hiwater_rss                   NaN
nr_ptes                       NaN
lock                          NaN
cgtime                        NaN
signal_nvcsw                  NaN
Name: classification, dtype: float64
```

- ***x=data1.drop(["hash","classification",'vm_truncate_count','shared_vm', 'exec_vm','nvcsw','maj_flt','utime'],axis=1)*** penjelasannya adalah Pertama, data1 merupakan sebuah dataframe atau struktur data yang

berisi beberapa kolom, seperti "hash", "classification", "vm_truncate_count", "shared_vm", "exec_vm", "nvcsw", "maj_flt", dan "utime".Kode data1.drop(["hash","classification",'vm_truncate_count','shared_vm','exec_vm','nvcsw','maj_flt','utime'],axis=1) digunakan untuk menghilangkan kolom-kolom tersebut dari dataframe data1. Parameter axis=1 menunjukkan bahwa yang akan dihapus adalah kolom, bukan baris. Dengan demikian, variabel x akan berisi dataframe baru yang terdiri dari kolom-kolom yang tersisa setelah dihapus.

```
x=data1.drop(["hash","classification",'vm_truncate_count','shared_vm','exec_vm','nvcsw','maj_flt','utime'],axis=1)
x.head()
```

	millisecond	state	usage_counter	prio	static_prio	normal_prio	policy	vm_pgoff	task_size	cached_hole_size	...	end
0	0	0	0	3069378560	14274	0	0	0	0	0	...	
1	1	0	0	3069378560	14274	0	0	0	0	0	...	
2	2	0	0	3069378560	14274	0	0	0	0	0	...	
3	3	0	0	3069378560	14274	0	0	0	0	0	...	
4	4	0	0	3069378560	14274	0	0	0	0	0	...	

5 rows × 27 columns

- y=data1["classification"] untuk menampung data1 dengan value ‘classification’ ke variabel y

```
y=data1["classification"]
y
```

```
0        1
1        1
2        1
3        1
4        1
       ..
99995    1
99996    1
99997    1
99998    1
99999    1
Name: classification, Length: 100000, dtype: int64
```

d. Membuat Model dengan Algoritma Naive Bayes

- Import library .

```
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
```

- Membagi data menjadi data latih dan data uji.

```
[32] x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=1)
```

- Membuat model pelatihan dengan algoritma Naive bayes.

```
from sklearn.naive_bayes import GaussianNB
model=GaussianNB()
model.fit(x_train,y_train)
```

```
▾ GaussianNB
GaussianNB()
```

- Membuat prediksi dari data uji x_test.

```
pred=model.predict(x_test)
pred
```

```
array([1, 1, 1, ..., 1, 0, 1])
```

- Menghitung model score dari data uji yang menghasilkan 0.6274 . Pada algoritma Naive Bayes, model.score(x_test, y_test) digunakan untuk menghitung akurasi dari model yang dilatih. Akurasi adalah ukuran yang mengindikasikan seberapa baik model dapat memprediksi label yang benar pada data pengujian. Fungsi model.score(x_test, y_test) akan menggunakan model Naive Bayes yang telah dilatih sebelumnya untuk melakukan prediksi pada x_test, dan kemudian membandingkan prediksi yang dihasilkan dengan label yang sebenarnya dalam y_test. Akurasi akan dihitung sebagai persentase prediksi yang benar dari keseluruhan data pengujian. Dari hasil model.score(x_test, y_test) adalah 0.6274, itu berarti model Naive Bayes dengan dataset pengujian tersebut mampu memprediksi dengan benar 62% dari sampel dalam x_test berdasarkan label yang ada dalam y_test. Semakin tinggi skor akurasi, semakin baik kinerja model dalam memprediksi kelas yang benar.

```
model.score(x_test,y_test)
```

```
0.6274
```

- Menampilkan perbandingan antara actual value dengan nilai prediksi yang sudah dihasilkan.

result

	Actual_Value	Predict_Value
43660	0	1
87278	1	1
14317	0	1
81932	1	1
95321	1	1
...	...	...
994	1	1
42287	0	1
4967	0	1
47725	0	0
42348	0	1

30000 rows × 2 columns

- Untuk file hasil prediksi bisa dilihat di github

e. Membuat Model dengan Algoritma SVM (Support Vector Machine)

- Import library SVM

```
[18] from sklearn.model_selection import train_test_split
     from sklearn.svm import SVC
     from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

- Membagi data menjadi data latih dan data uji

```
[19] X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

- Membuat model SVM

```
model.fit(X_train, y_train)

SVC
SVC(kernel='linear')
```

- Melatih data model dengan data latih

```
[22] y_pred = model.predict(X_test)
```

- Memprediksi label dari data uji

```
[23] accuracy = accuracy_score(y_test, y_pred)
     precision = precision_score(y_test, y_pred)
     recall = recall_score(y_test, y_pred)
     f1 = f1_score(y_test, y_pred)

     # Menampilkan hasil evaluasi
     print("Accuracy:", accuracy)
     print("Precision:", precision)
     print("Recall:", recall)
     print("F1-score:", f1)

     Accuracy: 0.5015
     Precision: 0.5
     Recall: 0.9988966900702106
     F1-score: 0.6664213062098502
```

- mencetak laporan klasifikasi (classification report) berdasarkan hasil prediksi y_pred dan nilai sebenarnya y_test dari model atau algoritma klasifikasi. Laporan klasifikasi yang dicetak akan mencakup beberapa metrik evaluasi klasifikasi yang penting, termasuk akurasi, presisi, recall, dan f1-score untuk setiap kelas yang diidentifikasi. Ini memberikan informasi tentang kinerja model dalam memprediksi masing-masing kelas, serta rata-rata secara keseluruhan. Parameter yang digunakan dalam kode tersebut adalah sebagai berikut:

  y_test: Merupakan nilai target atau nilai sebenarnya dari data yang akan dibandingkan dengan prediksi. Ini biasanya berupa array atau serangkaian nilai yang menunjukkan kelas atau label yang sebenarnya.

  y_pred: Merupakan hasil prediksi dari model atau algoritma klasifikasi terhadap data yang diuji. Ini juga berupa array atau serangkaian nilai yang menunjukkan kelas atau label yang diprediksi.

target_names: Merupakan parameter opsional yang digunakan untuk memberikan nama-nama kelas atau label yang sesuai dengan data. Dalam contoh ini, nama-nama kelas yang diberikan adalah 'Benign' (tidak berbahaya) dan 'Malware' (program berbahaya).

```
from sklearn.metrics import classification_report, confusion_matrix
print(classification_report(y_test, y_pred, target_names=['Benign', 'Malware']))
```

```
              precision    recall  f1-score   support

      Benign       0.87      0.01      0.01     10030
     Malware       0.50      1.00      0.67      9970

    accuracy                           0.50     20000
   macro avg       0.68      0.50      0.34     20000
weighted avg       0.68      0.50      0.34     20000
```

4. **Kesimpulan dan evaluasi**

Algoritma Naive bayes dalam konteks deteksi malware dapat digunakan untuk mengklasifikasikan program-program berdasarkan fitur-fitur yang ada. Dalam model yang telah saya buat dimana dari pelatihan model menggunakan algoritma naive bayes yang menghasilkan score 0.6247 itu berarti model Naive Bayes dengan dataset pengujian tersebut mampu memprediksi dengan benar 62% dari sampel dalam x_test berdasarkan label yang ada dalam y_test. Semakin tinggi skor akurasi, semakin baik kinerja model dalam memprediksi kelas yang benar. Lalu dihasilkan perbandingan dari actual value yang berisi data dari label y_test dan nilai prediksi yang dihasilkan dari prediksi model yang dihasilkan.

Algoritma SVM adalah algoritma pembelajaran mesin yang digunakan untuk tugas klasifikasi dan regresi. Dalam konteks deteksi malware, SVM dapat digunakan untuk memisahkan program-program berbahaya dari program-program yang aman dengan mencari hyperplane pemisah yang optimal. Dalam model yang telah saya buat menghasilkan nilai precision , recall, f1-score, dan support dari masing-masing klasifikasi label ‘Benign’ dan ‘Malware’ yang telah diolah dari pemodelan SVM. dari nilai yang dihasilkan memberikan kinerja model dalam masing-masing kelas serta rata-rata dari keseluruhan.

- Precision (presisi) mengukur sejauh mana hasil positif yang diprediksi benar. Untuk kelas "Benign" (tidak berbahaya), presisi adalah 0,87, yang berarti sekitar 87% dari program yang diprediksi sebagai tidak berbahaya

memang benar-benar tidak berbahaya. Untuk kelas "Malware" (program berbahaya), presisi adalah 0,50, yang berarti sekitar 50% dari program yang diprediksi sebagai berbahaya memang benar-benar berbahaya.

- Recall (sensitivitas) mengukur sejauh mana program yang benar-benar berbahaya yang berhasil diidentifikasi oleh model. Untuk kelas "Benign", recall adalah 0,01, yang berarti hanya sekitar 1% dari program yang benar-benar tidak berbahaya yang berhasil diidentifikasi. Untuk kelas "Malware", recall adalah 1,00 yang berarti model berhasil mengidentifikasi semua program berbahaya dengan baik.
- F1-score merupakan ukuran gabungan dari presisi dan recall. F1-score untuk kelas "Benign" adalah 0,01 sementara untuk kelas "Malware" adalah 0,67.
- Support menunjukkan jumlah sampel dalam set pengujian untuk setiap kelas.
- Accuracy (akurasi) adalah persentase dari prediksi yang benar secara keseluruhan. Dalam laporan ini, akurasi adalah 0,50, yang berarti model mengklasifikasikan dengan benar sekitar 50% dari total sampel.
- Macro avg adalah nilai rata-rata dari presisi, recall, dan f1-score untuk semua kelas.
- Weighted avg adalah rata-rata yang tumpang tindih, di mana setiap kelas dihitung berdasarkan bobotnya dalam jumlah sampel.

LINK Github CODE :
https://github.com/kopibalap/Malware-Detection---Machine-Learning.git

DAFTAR PUSTAKA

- https://sis.binus.ac.id/2022/02/14/support-vector-machine-algorithm/
- https://binus.ac.id/bandung/2019/12/algoritma-naive-bayes/
- https://www.kaggle.com/code/nsaravana/malware-detection-using-naive-bayes
- https://www.kaggle.com/code/maidaly/malware-detection-with-machine-learning/notebook