ERA AI

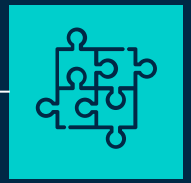
Alifah Azka Nisrina - 1906353662

Muhammad Irza Arrizkyputra - 1906353744

Bonifasius Erlangga Einsoni Rienko - 1906302850

Andrew Nehemia H - 1906400311

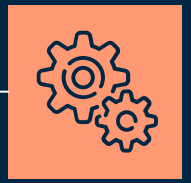
TABLE OF CONTENTS



01

Dataset

Mengenai kondisi dataset



02

EDA

Exploratory Data Analysis

03

Data Preprocessing

Data preparation

TABLE OF CONTENTS

04

Classification

- Decision Tree
- Random Forest
- Naive Bayes
- KNN

05

Regression

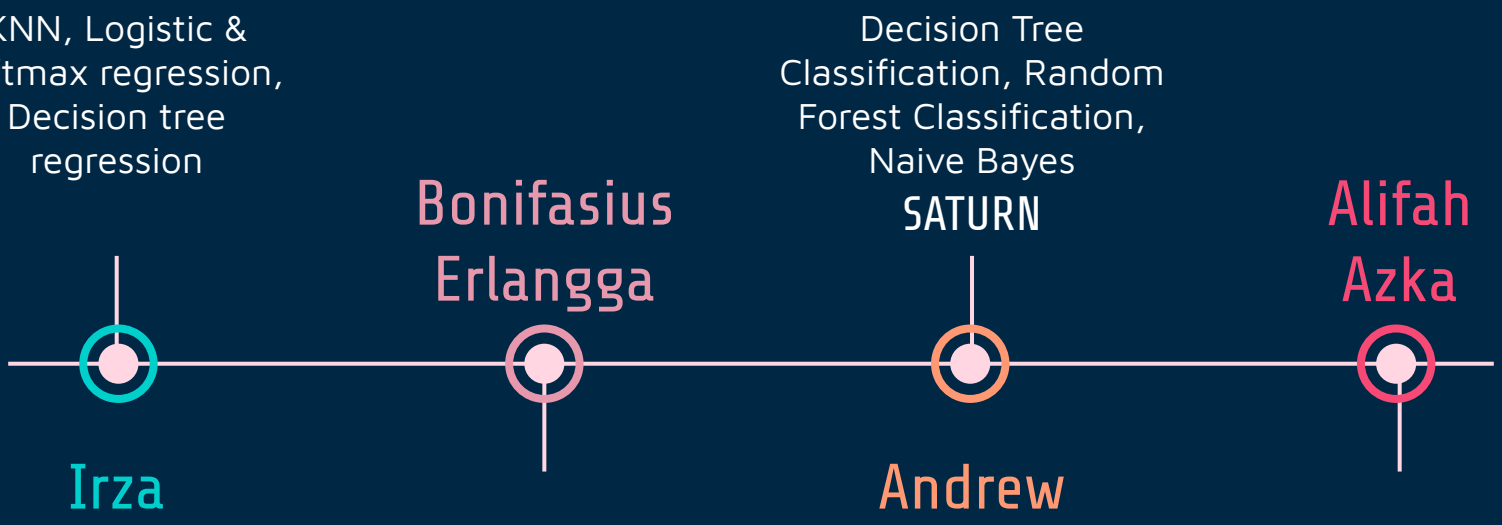
- Logistic Regression
- Softmax Regression
- Random Forest
- Linear Regression
- Lasso
- Decision Tree
- Ridge

06

Clustering

- K-Means
- Hierarchical

PEMBAGIAN TUGAS



DATASET

01

```
dv = pd.read_csv("09_DeteksiVirus.csv")

print("Data has size:", dv.shape)
dv.head()
```

```
Data has size: (149668, 44)
/usr/local/lib/python3.8/dist-packages/IPython/core/interactiveshell.py:3326: DtypeWarning: Columns (27) have mixed types.Specify dtype option on import or set low_mem
  exec(code_obj, self.user_global_ns, self.user_ns)
```

	IdDefaultBrowser	IdSettingAntivirus	BanyakAntivirus	IdNegaraPembuat	IdKotaPembuat	IdOrganisasiPembuat	IdLokasiGeografisMesinSaatIni	Platform	Processor	OsSu
0	NaN	46413.0	2.0	68	151854.0	27.0	276.0	windows10	x64	
1	NaN	5106.0	3.0	57	117751.0	NaN	277.0	windows10	x64	
2	NaN	53447.0	1.0	93	36825.0	NaN	119.0	windows10	x64	
3	NaN	53447.0	1.0	50	115291.0	NaN	98.0	windows10	x86	
4	3176.0	7945.0	2.0	68	43129.0	27.0	150.0	windows10	x64	

5 rows × 44 columns

Dataset tersebut memiliki 44 atribut dan 149668 baris

#	Column	Non-Null Count	Dtype
0	IdDefaultBrowser	4613 non-null	float64
1	IdSettingAntivirus	134400 non-null	float64
2	BanyakAntivirus	134400 non-null	float64
3	IdNegaraPembuat	149668 non-null	int64
4	IdKotaPembuat	144598 non-null	float64
5	IdOrganisasiPembuat	105521 non-null	float64
6	IdLokasiGeografisMesinSaatIni	149646 non-null	float64
7	Platform	149668 non-null	object
8	Processor	149668 non-null	object
9	OsSuite	149668 non-null	int64
10	OsPlatformSubRelease	149668 non-null	object
11	VersiInternetExplorer	148834 non-null	float64
12	SmartScreenSetting	87660 non-null	object
13	DeviceType	149668 non-null	object
14	IdOEM	148392 non-null	float64
15	IdModelOEM	148310 non-null	float64
16	BanyakCoreProcessor	148638 non-null	float64
17	IdPembuatProcessor	148638 non-null	float64
18	IdModelProcessor	148636 non-null	float64
19	KapasitasDiskMemory	149231 non-null	float64
20	TipeDiskUtama	144981 non-null	object
21	KapasitasVolumeSistem	149231 non-null	float64
22	KapasitasRAM	147415 non-null	float64
23	TipeChassis	149430 non-null	object
24	UkuranDiagonalLayar	143543 non-null	float64
25	UkuranHorisontalLayar	143547 non-null	float64
26	UkuranVertikalLayar	143547 non-null	float64
27	TipeBateraiInternal	54931 non-null	object
28	VersiOS	149668 non-null	object
29	ArsitekturOS	149668 non-null	object
30	BranchOS	149668 non-null	object
31	BuildOS	149668 non-null	int64
32	RevisiBuildOS	149668 non-null	int64
33	EdisiOS	149668 non-null	object
34	SkuNameOS	149668 non-null	object
35	TipeInstallasiOS	149668 non-null	object
36	AutoUpdateSetting	149668 non-null	object
37	IsOSGenuine	149668 non-null	object
38	IdPembuatFirmware	145604 non-null	float64
39	IdVersiFirmware	145763 non-null	float64
40	IsSecureBootEnabled	149668 non-null	int64
41	IsTouchScreen	149668 non-null	int64
42	IsGamer	148978 non-null	float64
43	infected_proba	149668 non-null	float64

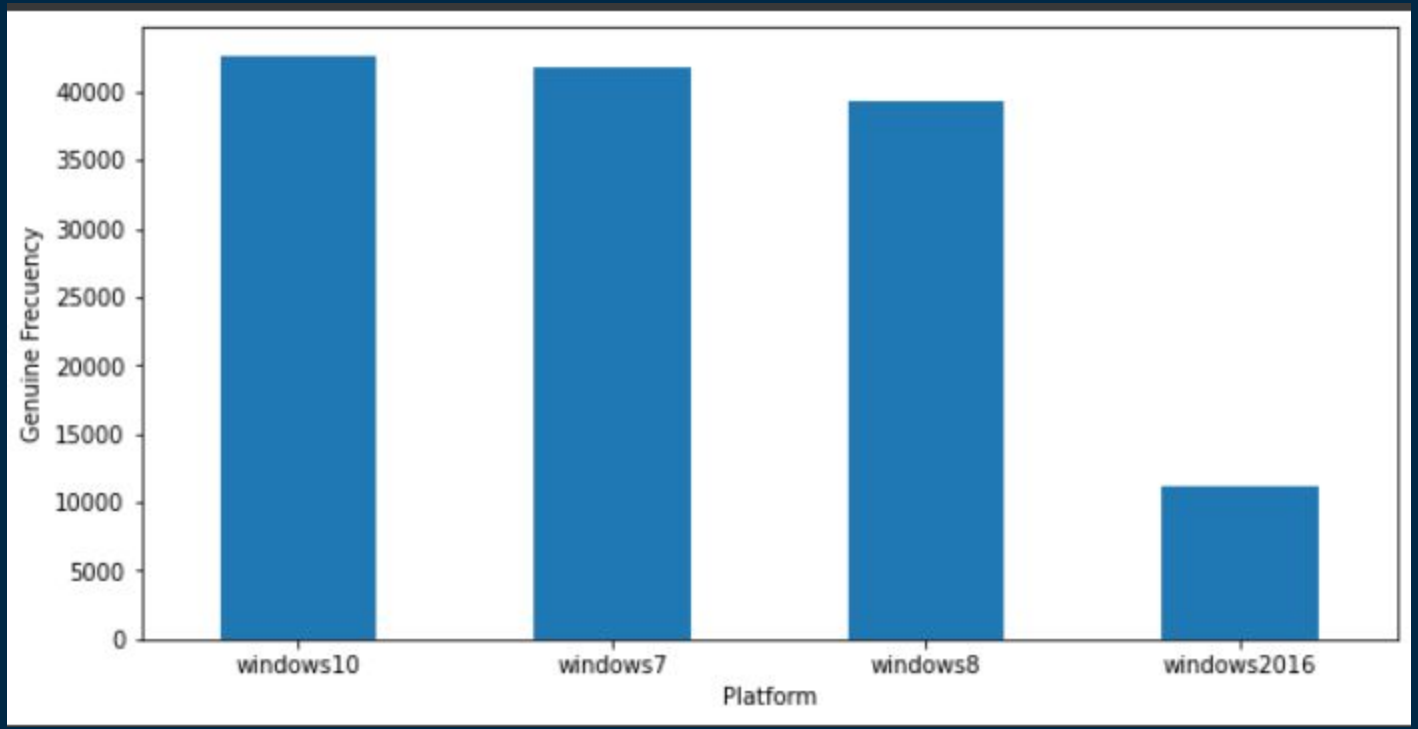
dtypes: float64(22), int64(6), object(16)
memory usage: 50.2+ MB

Dataset tersebut terdapat 22 data tipe data float, 6 tipe data integer, dan 16 tipe data

Exploratory Data Analysis (EDA)

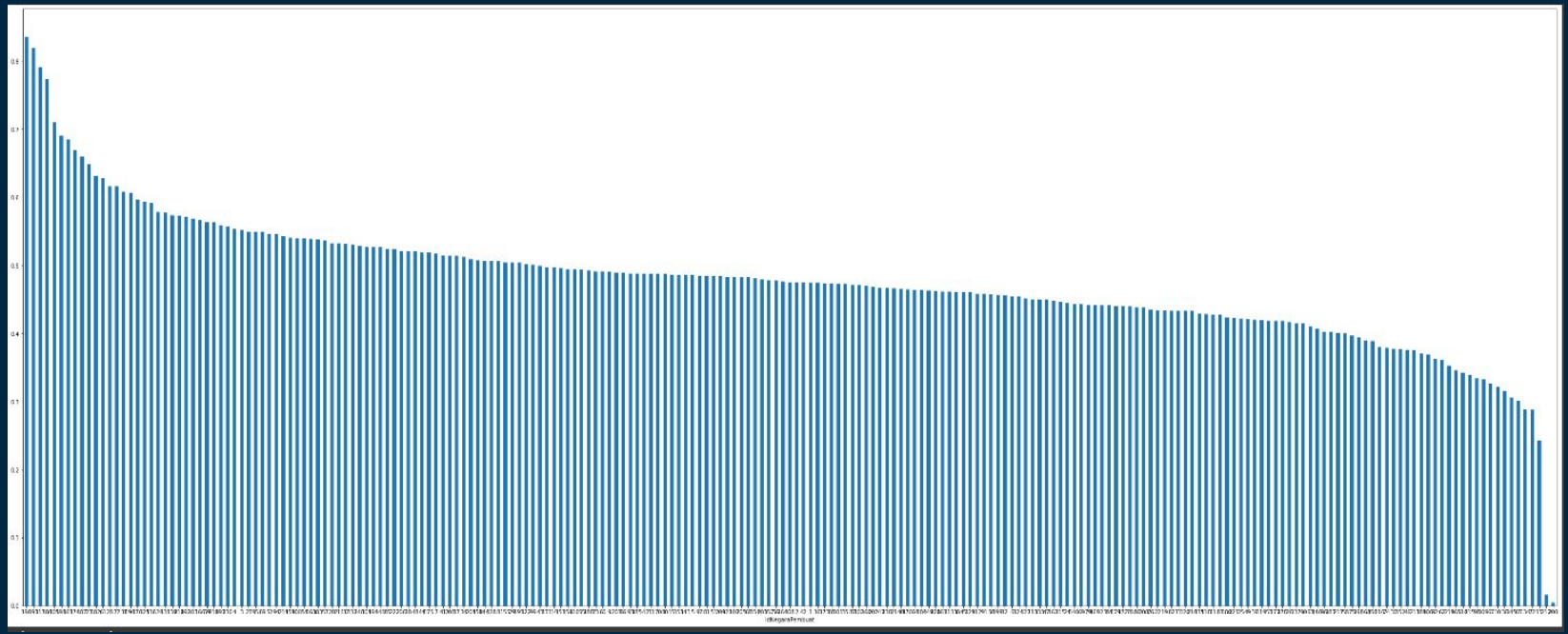
02

Hubungan Platform dengan Mesin Software



Windows10 merupakan platform yang memiliki jumlah rata-rata OS asli terbanyak

Negara Produsen dengan Ketahanan Virus



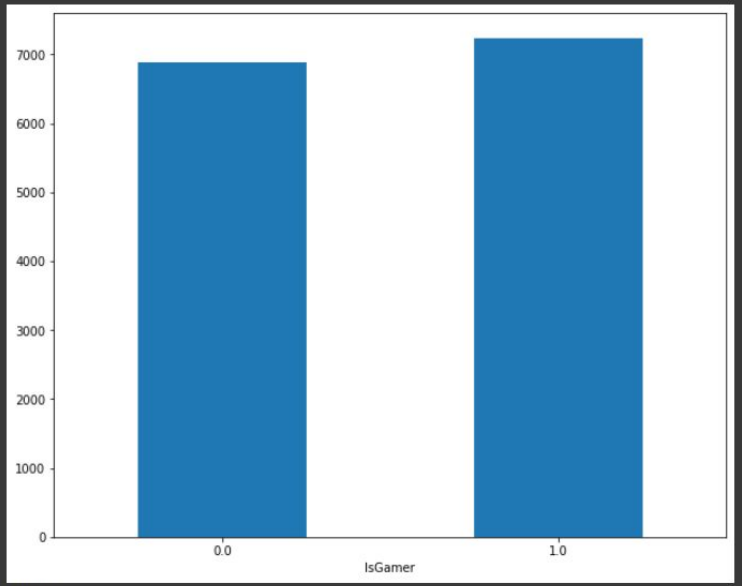
Negara Produsen dengan Ketahanan Virus

```
IdNegaraPembuat
196    0.835667
193    0.820000
117    0.790692
186    0.774000
105    0.710375
          ...
134    0.289125
72     0.287909
13     0.243111
212    0.016000
200    0.005000
Name: infected_proba, Length: 221, dtype: float64
```

```
134     8
72     11
13     18
212     1
200     1
Name: IdNegaraPembuat, dtype: int64
```

Di atas merupakan 5 id negara pembuat antivirus yang paling rentan mesinnya terkena virus dan 5 id negara pembuat antivirus yang paling sedikit probabilitas mesinnya terkena virus. Berdasarkan jumlah dari masing-masing atribut, mungkin dapat dikatakan untuk id 200 dan 212 kurang valid karena jumlahnya hanya 1. Negara dengan id 13 memiliki kemunculan lebih sering dan probabilitas terinfeksi virusnya juga cenderung masih kecil

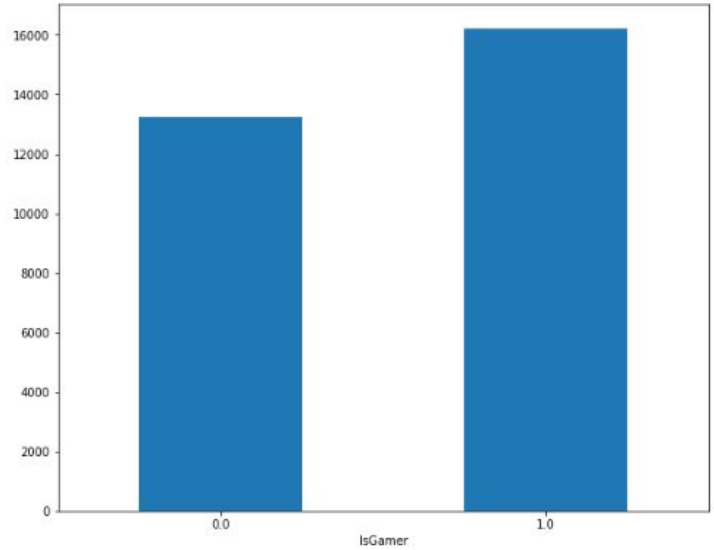
Perbedaan Mesin Gaming




```
IsGamer
0.0    6890.443821
1.0    7242.459130
Name: KapasitasRAM, dtype: float64
```

Mesin Gaming memiliki Kapasitas RAM yang lebih besar

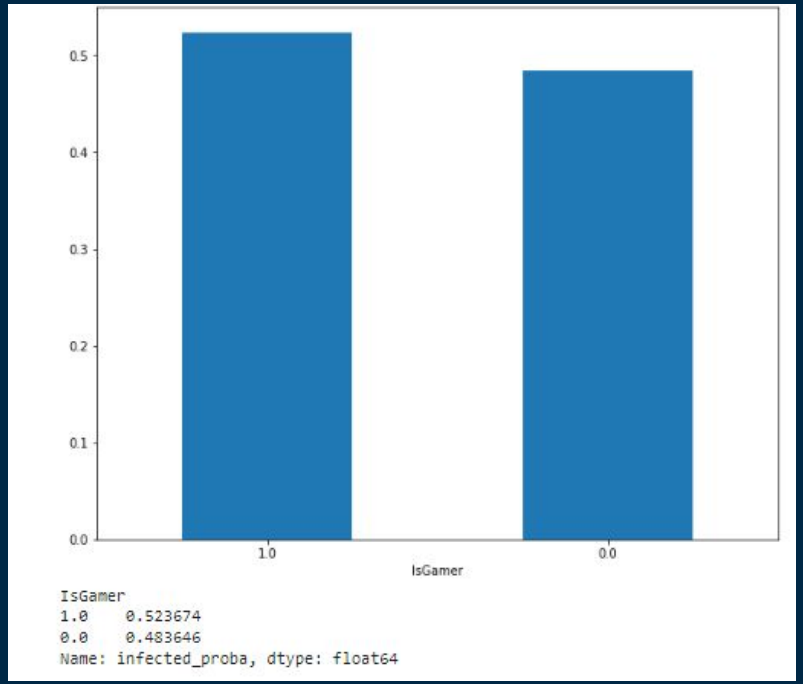
Perbedaan Mesin Gaming




```
IsGamer
0.0    13232.276126
1.0    16227.513230
Name: BuildOS, dtype: float64
```

Mesin Gaming memiliki BuildOS lebih tinggi

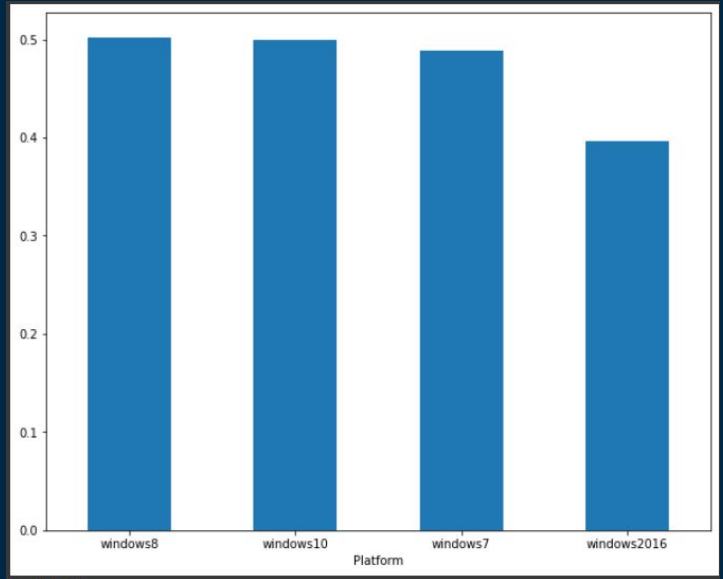
Perbedaan Mesin Gaming




```
IsGamer
1.0    0.523674
0.0    0.483646
Name: infected_proba, dtype: float64
```

Mesin Gaming memiliki kemungkinan terinfeksi virus lebih tinggi

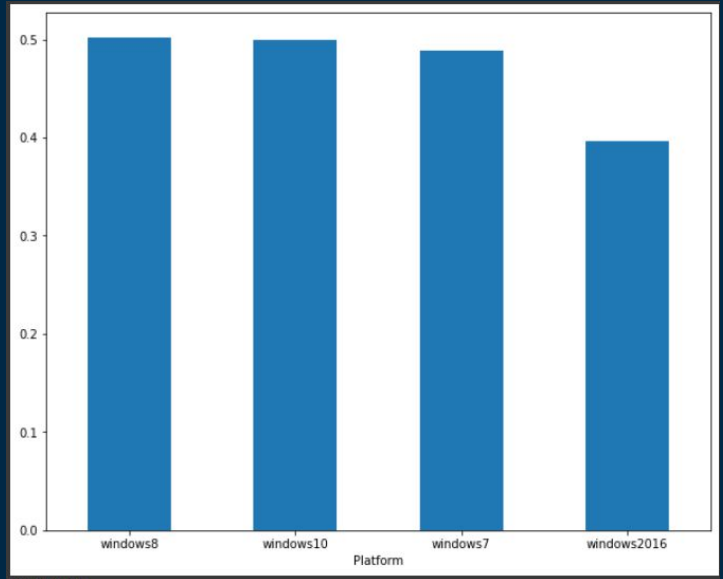
Perbedaan Software terhadap Kerentanan Mesin




```
Platform
windows8       0.502410
windows10      0.500041
windows7       0.488928
windows2016    0.396228
Name: infected_proba, dtype: float64
```

Dari dataset yang dimiliki, Windows8 yang paling rentan terkena virus

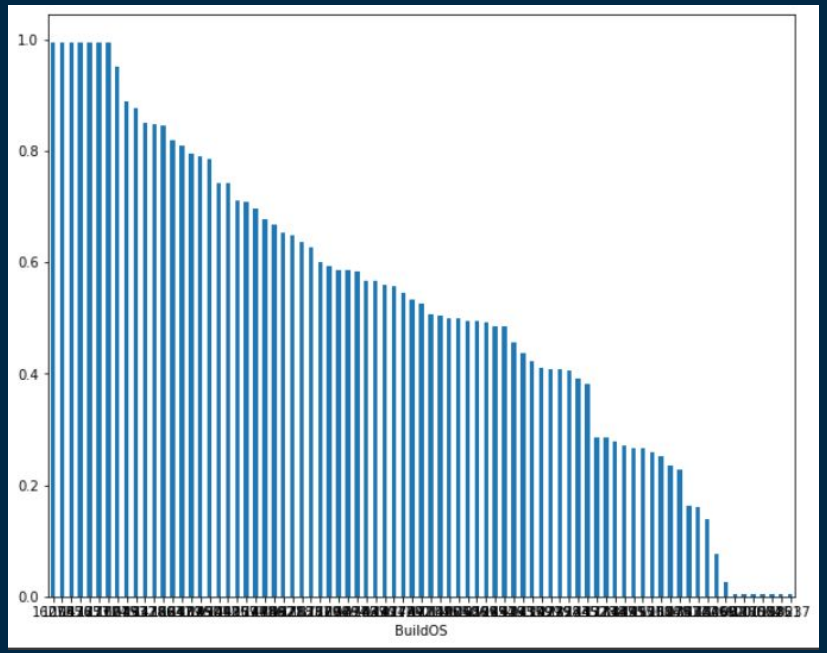
Perbedaan Software terhadap Kerentanan Mesin




```
Platform
windows8       0.502410
windows10      0.500041
windows7       0.488928
windows2016    0.396228
Name: infected_proba, dtype: float64
```

Dari dataset yang dimiliki, Windows8 yang paling rentan terkena virus

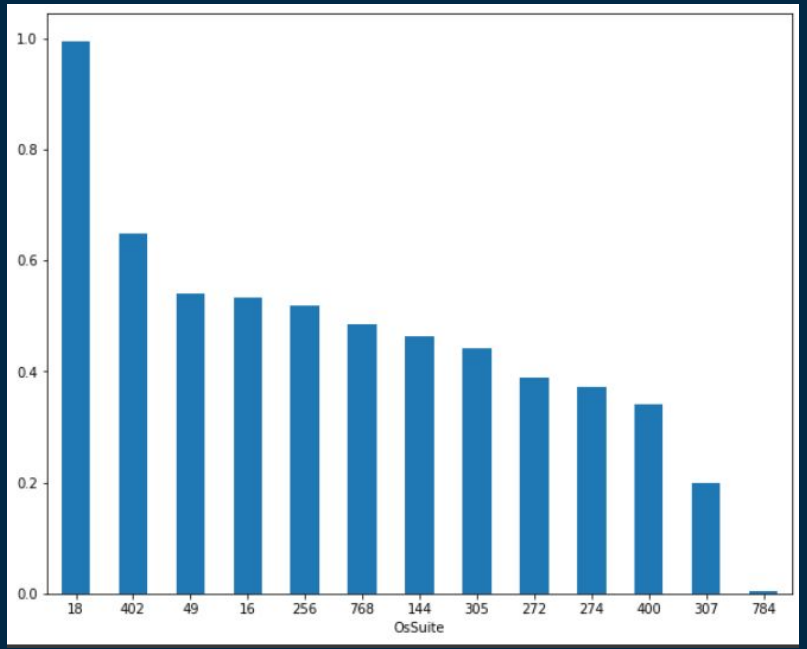
Perbedaan Software terhadap Kerentanan Mesin




```
BuildOS
16273    0.995
10565    0.995
10576    0.995
14385    0.995
17733    0.995
         ...
17666    0.005
17074    0.005
15025    0.005
15061    0.005
18237    0.005
Name: infected_proba, Length: 81, dtype: float64
```

Dari dataset yang dimiliki, Build OS 16273 yang paling rentan terkena virus

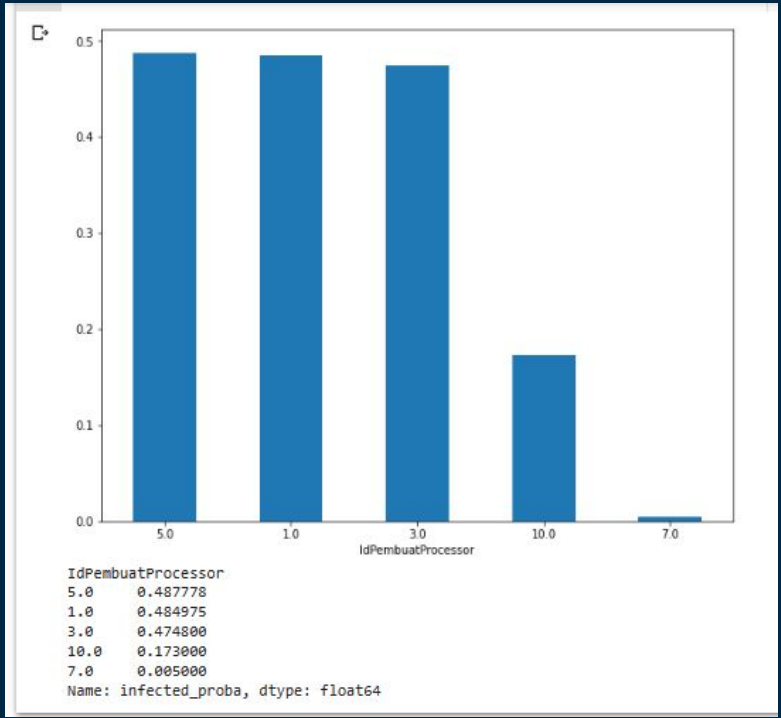
Perbedaan Software terhadap Kerentanan Mesin




```
OsSuite
18     0.995000
402    0.649000
49     0.540529
16     0.531836
256    0.518255
768    0.484604
144    0.462618
305    0.442178
272    0.388840
274    0.371359
400    0.340218
307    0.199500
784    0.005000
Name: infected_proba, dtype: float64
```

Dari dataset yang dimiliki, OS Suite 18 yang paling rentan terkena virus

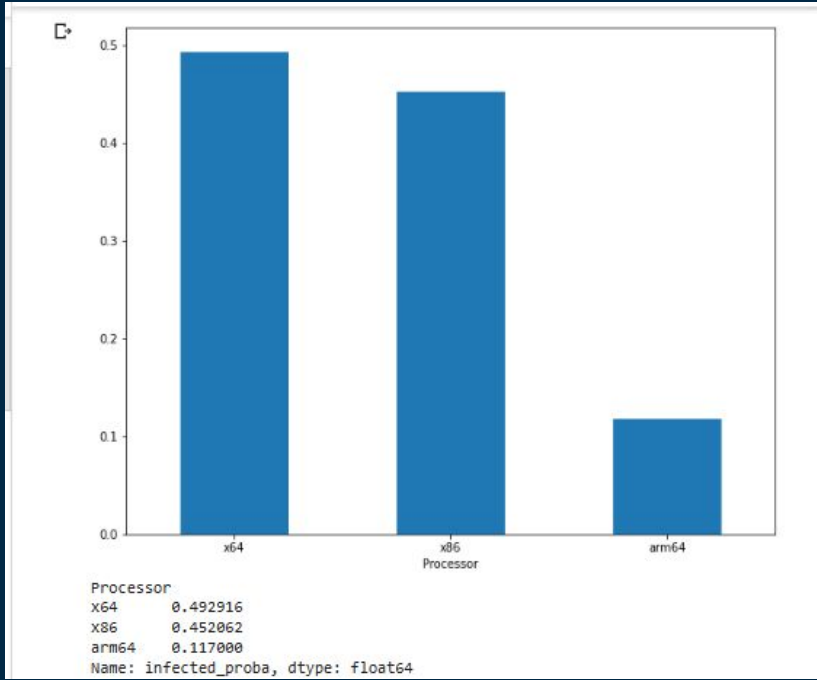
Perbedaan Hubungan Hardware terhadap Kerentanan Mesin




```
IdPembuatProcessor
5.0     0.487778
1.0     0.484975
3.0     0.474800
10.0    0.173000
7.0     0.005000
Name: infected_proba, dtype: float64
```

Pembuat prosesor bisa dianggap memiliki kerentanan yang sama terhadap virus. Id 7 dan 10 bisa dianggap kurang valid karena jumlah rownya kurang dari tiga

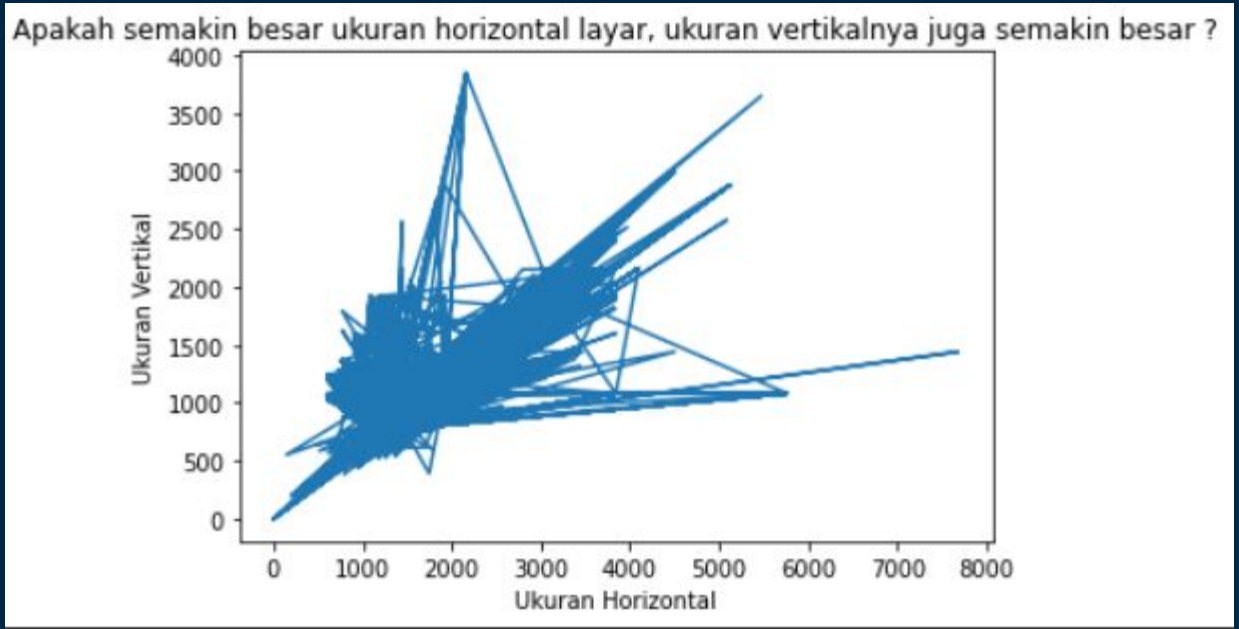
Perbedaan Hubungan Hardware terhadap Kerentanan Mesin




```
Processor
x64      0.492916
x86      0.452062
arm64    0.117000
Name: infected_proba, dtype: float64
```

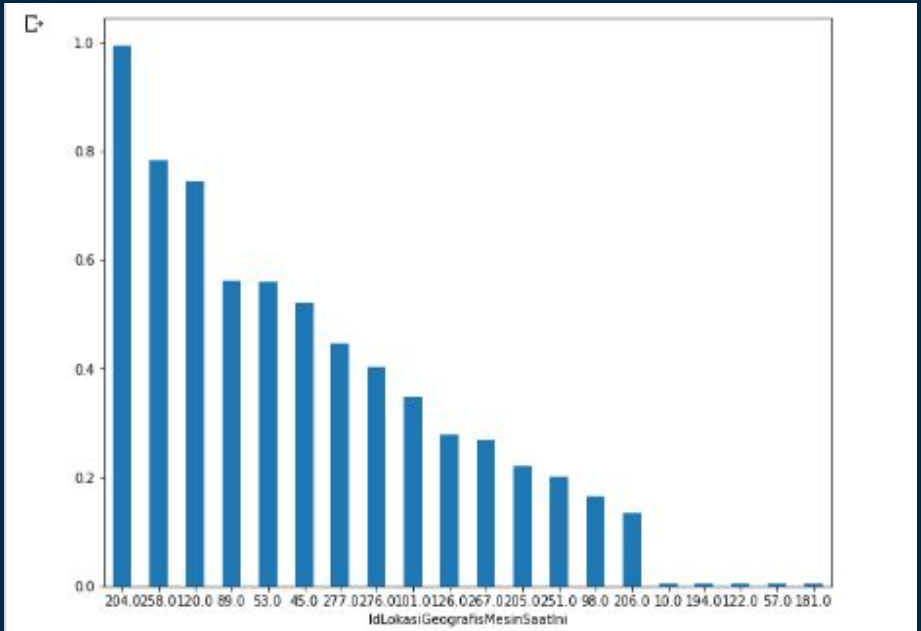
Setiap prosesor bisa dianggap memiliki kerentanan yang sama terhadap virus. Prosesor arm64 dianggap kurang valid karena jumlah rownya kurang dari tiga

Korelasi Ukuran Layar



Semakin besar ukuran layar secara horizontal, ukuran vertikal juga semakin besar

Ketahanan virus berdasarkan lokasi mesin



IdLokasiGeografisMesinSaatIn	
204.0	0.995000
258.0	0.782000
120.0	0.744000
89.0	0.561833
53.0	0.560000
45.0	0.520000
277.0	0.445857
276.0	0.402000
101.0	0.348500
126.0	0.278000
267.0	0.268000
205.0	0.220000
251.0	0.202000
98.0	0.165000
206.0	0.134000
10.0	0.005000
194.0	0.005000
122.0	0.005000
57.0	0.005000
181.0	0.005000

Data yang ditampilkan adalah LokasiMesin dengan *occurence* lebih dari 1000. Dapat dikatakan negara 204 99 persen warganya terinfeksi virus. Sedangkan pada negara 10,194,122,57,181, dan lainnya sangat rendah mesinnya terinfeksi virus

Data Preprocessing

03

Duplicated Data

```
[ ]  dv_cleaned = dv.copy()
```

```
[ ]  dv_cleaned.duplicated(keep=False).sum()
```

```
     101
```

```
[ ]  # dropping duplicate values
     dv_cleaned = dv_cleaned.drop_duplicates()

     dv_cleaned.duplicated(keep=False).sum()
```

```
     0
```

Outliers

```
def outlier_counter(df):
  q1 = df.quantile(q = 0.25)
  q3 = df.quantile(q = 0.75)
  iqr = q3 - q1
  hasil = ((df < (q1 - 1.5 * iqr)) | (df > (q3 + 1.5 * iqr))).sum()
  return hasil
```

Membuat function untuk menghitung jumlah outlier

Outliers

```
[ ]  def outlier_to_upper_lower(df, column):
       # membuat outliers menjadi NaN Value
       quartile1 = df[column].quantile(q=0.25)
       quartile3 = df[column].quantile(q=0.75)
       iqr = quartile3 - quartile1
       lower = quartile1 - 1.5 * iqr
       upper = quartile3 + 1.5 * iqr
       df.loc [df[column] < (quartile1 - 1.5 * iqr), column] = lower
       df.loc [df[column] > (quartile3 + 1.5 * iqr), column] = upper
       # memeriksa apakah masih ada outliers atau tidak
       quartile1 = df[column].quantile(q=0.25)
       quartile3 = df[column].quantile(q=0.75)
       recent_outlier = ((df[column] < (quartile1 - 1.5 * iqr)) | (df[column] > (quartile3 + 1.5 * iqr))).sum()
       print("outlier saat ini ada sebanyak",recent_outlier)
```

```
[ ]  def outlier_info(df, column):
       q1 = df[column].quantile(q = 0.25)
       q3 = df[column].quantile(q = 0.75)
       iqr = q3 - q1
       lower = q1 - 1.5 * iqr
       upper = q3 + 1.5 * iqr
       median = df[column].median()
       print("q1: "+str(q1))
       print("q3: "+str(q3))
       print("lower: "+str(lower))
       print("upper: "+str(upper))
       print("median: "+str(median))
```

Membuat function untuk menangani outlier dengan cara mengubah menjadi upper atau menjadi lower, dan function untuk melihat informasi outlier seperti upper/lower limit serta mediannya

Outliers

ArsitekturOS	0
AutoUpdateSetting	0
BanyakAntivirus	94
BanyakCoreProcessor	16220
BranchOS	0
BuildOS	0
DeviceType	0
EdisiOS	0
IdDefaultBrowser	0
IdKotaPembuat	0
IdLokasiGeografisMesinSaatIni	0
IdModelOEM	1592
IdModelProcessor	10582
IdNegaraPembuat	0
IdOEM	1233
IdOrganisasiPembuat	2210
IdPembuatFirmware	0
IdPembuatProcessor	18765
IdSettingAntivirus	0
IdVersiFirmware	0
IsGamer	15135
IsOSGenuine	0
IsSecureBootEnabled	0
IsTouchScreen	14531
KapasitasDiskMemory	3183
KapasitasRAM	11677
KapasitasVolumeSistem	2073
OsPlatformSubRelease	0
OsSuite	0
Platform	0
Processor	0
RevisiBuildOS	20404
SkuNameOS	0
SmartScreenSetting	0
TipeBateraiInternal	0
TipeChassis	0
TipeDiskUtama	0
TipeInstallasiOS	0

IdDefaultBrowser	0
IdKotaPembuat	0
IdLokasiGeografisMesinSaatIni	0
IdModelOEM	1592
IdModelProcessor	10582
IdNegaraPembuat	0
IdOEM	1233
IdOrganisasiPembuat	2210
IdPembuatFirmware	0
IdPembuatProcessor	18765
IdSettingAntivirus	0
IdVersiFirmware	0
IsGamer	15135
IsOSGenuine	0
IsSecureBootEnabled	0
IsTouchScreen	14531
KapasitasDiskMemory	3183
KapasitasRAM	11677
KapasitasVolumeSistem	2073
OsPlatformSubRelease	0
OsSuite	0
Platform	0
Processor	0
RevisiBuildOS	20404
SkuNameOS	0
SmartScreenSetting	0
TipeBateraiInternal	0
TipeChassis	0
TipeDiskUtama	0
TipeInstallasiOS	0
UkuranDiagonalLayar	4267
UkuranHorisontalLayar	3453
UkuranVertikalLayar	2409
VersiInternetExplorer	0
VersiOS	0
infected_proba	0

Berikut adalah jumlah outlier yang ada pada masing-masing atribut

Outliers

```
dv_cleaned['IsGamer'].value_counts()
```

```
0.0    133780
1.0     15135
Name: IsGamer, dtype: int64
```

```
dv_cleaned['IsTouchScreen'].value_counts()
```

```
0    135074
1     14531
Name: IsTouchScreen, dtype: int64
```

Untuk outlier IsGamer dan IsTouchScreen menurut kami tidak perlu ditangani karena sebenarnya valuenya hanya 2, yaitu benar (1) atau tidak (0)

Outliers

```
outlier_to_upper_lower(dv_cleaned,"BanyakCoreProcessor")
outlier saat ini ada sebanyak 0

outlier_to_upper_lower(dv_cleaned,"RevisiBuildOS")
outlier saat ini ada sebanyak 0
```

```
[ ] outlier_to_upper_lower(dv_cleaned,"BanyakAntivirus")
    outlier saat ini ada sebanyak 0

[ ] outlier_to_upper_lower(dv_cleaned,"KapasitasDiskMemory")
    outlier saat ini ada sebanyak 0

[ ] outlier_to_upper_lower(dv_cleaned,"KapasitasRAM")
    outlier saat ini ada sebanyak 0

[ ] outlier_to_upper_lower(dv_cleaned,"KapasitasVolumeSistem")
    outlier saat ini ada sebanyak 0

    outlier_to_upper_lower(dv_cleaned,"UkuranDiagonalLayar")
    outlier saat ini ada sebanyak 0

[ ] outlier_to_upper_lower(dv_cleaned,"UkuranHorisontalLayar")
    outlier saat ini ada sebanyak 0

[ ] outlier_to_upper_lower(dv_cleaned,"UkuranVertikalLayar")
    outlier saat ini ada sebanyak 0
```

Outliers

ArsitekturOS	0
AutoUpdateSetting	0
BanyakAntivirus	0
BanyakCoreProcessor	0
BranchOS	0
BuildOS	0
DeviceType	0
EdisiOS	0
IdDefaultBrowser	0
IdKotaPembuat	0
IdLokasiGeografisMesinSaatIni	0
IdModelOEM	1592
IdModelProcessor	10582
IdNegaraPembuat	0
IdOEM	1233
IdOrganisasiPembuat	2210
IdPembuatFirmware	0
IdPembuatProcessor	18765
IdSettingAntivirus	0
IdVersiFirmware	0
IsGamer	15135
IsOSGenuine	0
IsSecureBootEnabled	0
IsTouchScreen	14531
KapasitasDiskMemory	0
KapasitasRAM	0
KapasitasVolumeSistem	0
OsPlatformSubRelease	0
OsSuite	0
Platform	0
Processor	0
RevisiBuildOS	0
SkuNameOS	0
SmartScreenSetting	0
TipeBateraiInternal	0
TipeChassis	0
TipeDiskUtama	0
TipeInstallasiOS	0

EdisiOS	0
IdDefaultBrowser	0
IdKotaPembuat	0
IdLokasiGeografisMesinSaatIni	0
IdModelOEM	1592
IdModelProcessor	10582
IdNegaraPembuat	0
IdOEM	1233
IdOrganisasiPembuat	2210
IdPembuatFirmware	0
IdPembuatProcessor	18765
IdSettingAntivirus	0
IdVersiFirmware	0
IsGamer	15135
IsOSGenuine	0
IsSecureBootEnabled	0
IsTouchScreen	14531
KapasitasDiskMemory	0
KapasitasRAM	0
KapasitasVolumeSistem	0
OsPlatformSubRelease	0
OsSuite	0
Platform	0
Processor	0
RevisiBuildOS	0
SkuNameOS	0
SmartScreenSetting	0
TipeBateraiInternal	0
TipeChassis	0
TipeDiskUtama	0
TipeInstallasiOS	0
UkuranDiagonalLayar	0
UkuranHorisontalLayar	0
UkuranVertikalLayar	0
VersiInternetExplorer	0
VersiOS	0
infected_proba	0
dtype: int64	

Berikut adalah jumlah outlier yang ada pada masing-masing atribut setelah dilakukan penanganan outlier

Terdapat outlier yang tidak ditangani yaitu atribut-atribut yang sifatnya Id dan atribut yang nilainya binary value (hanya 1 atau 0)

Missing Value

```python
def cek_null(df):
    col_na = df.isnull().sum().sort_values(ascending=False)
    percent = col_na / len(df)

    missing_data = pd.concat([col_na, percent], axis=1, keys=['Total', 'Percent'])
    print(missing_data[missing_data['Total'] > 0])
```

```python
cek_null(dv_cleaned)
```

	Total	Percent
IdDefaultBrowser	144992	0.969165
TipeBateraiInternal	94699	0.632994
SmartScreenSetting	61987	0.414338
IdOrganisasiPembuat	44123	0.294930
BanyakAntivirus	15235	0.101835
IdSettingAntivirus	15235	0.101835
UkuranDiagonalLayar	6120	0.040908
UkuranVertikalLayar	6116	0.040881
UkuranHorisontalLayar	6116	0.040881
IdKotaPembuat	5067	0.033869
TipeDiskUtama	4687	0.031329
IdPembuatFirmware	4061	0.027145
IdVersiFirmware	3902	0.026082
KapasitasRAM	2249	0.015033
IdModelOEM	1358	0.009077
IdOEM	1276	0.008529
IdModelProcessor	1032	0.006898
BanyakCoreProcessor	1030	0.006885
IdPembuatProcessor	1030	0.006885
VersiInternetExplorer	834	0.005575
IsGamer	690	0.004612
KapasitasDiskMemory	437	0.002921
KapasitasVolumeSistem	437	0.002921
TipeChassis	238	0.001591
IdLokasiGeografisMesinSaatIni	22	0.000147

Memeriksa jumlah missing value setiap atribut

Missing Value

```
dv_cleaned.drop('IdDefaultBrowser', inplace=True, axis=1)
```

```
[ ] dv_cleaned['IdLokasiGeografisMesinSaatIni'].fillna(value = 0, inplace = True)

[ ] dv_cleaned['IdModelProcessor'].fillna(value = 0, inplace = True)

[ ] dv_cleaned['IdOEM'].fillna(value = 0, inplace = True)

[ ] dv_cleaned['IdModelOEM'].fillna(value = 0, inplace = True)

[ ] dv_cleaned['IdVersiFirmware'].fillna(value = 0, inplace = True)

[ ] dv_cleaned['IdPembuatFirmware'].fillna(value = 0, inplace = True)

[ ] dv_cleaned.drop('IdSettingAntivirus', inplace=True, axis=1)

[ ] dv_cleaned.drop('IdOrganisasiPembuat', inplace=True, axis=1)

[ ] dv_cleaned['IdKotaPembuat'].fillna(value = 0, inplace = True)

[ ] dv_cleaned['IdPembuatProcessor'].fillna(value = 0, inplace = True)

dv_cleaned['TipeBateraiInternal'] = dv_cleaned['TipeBateraiInternal'].fillna(dv_cleaned['TipeBateraiInternal'].mode()[0])

[ ] dv_cleaned['SmartScreenSetting'] = dv_cleaned['SmartScreenSetting'].fillna(dv_cleaned['SmartScreenSetting'].mode()[0])

[ ] dv_cleaned['BanyakAntivirus'] = dv_cleaned['BanyakAntivirus'].fillna(dv_cleaned['BanyakAntivirus'].median())
```

Untuk penanganannya, IdDefaultBrowser kami drop selain karena sifatnya Id yang dimana sepertinya tidak dibutuhkan di langkah-langkah berikutnya, juga karena missing valuenya mencapai kurang lebih 97%.

Sisanya, penanganannya adalah dengan mengisi missing value dengan nilai median atau modusnya

Missing Value

```
[ ] dv_cleaned['BanyakAntivirus'] = dv_cleaned['BanyakAntivirus'].fillna(dv_cleaned['BanyakAntivirus'].median())

[ ] dv_cleaned['UkuranDiagonalLayar'] = dv_cleaned['UkuranDiagonalLayar'].fillna(dv_cleaned['UkuranDiagonalLayar'].median())

[ ] dv_cleaned['UkuranVertikalLayar'] = dv_cleaned['UkuranVertikalLayar'].fillna(dv_cleaned['UkuranVertikalLayar'].median())

[ ] dv_cleaned['UkuranHorisontalLayar'] = dv_cleaned['UkuranHorisontalLayar'].fillna(dv_cleaned['UkuranHorisontalLayar'].median())

[ ] dv_cleaned['TipeDiskUtama'] = dv_cleaned['TipeDiskUtama'].fillna(dv_cleaned['TipeDiskUtama'].mode()[0])

[ ] dv_cleaned['KapasitasRAM'] = dv_cleaned['KapasitasRAM'].fillna(dv_cleaned['KapasitasRAM'].median())

[ ] dv_cleaned['BanyakCoreProcessor'] = dv_cleaned['BanyakCoreProcessor'].fillna(dv_cleaned['BanyakCoreProcessor'].median())

[ ] dv_cleaned['VersiInternetExplorer'] = dv_cleaned['VersiInternetExplorer'].fillna(dv_cleaned['VersiInternetExplorer'].median())

[ ] dv_cleaned['IsGamer'] = dv_cleaned['IsGamer'].fillna(dv_cleaned['IsGamer'].median())

[ ] dv_cleaned['KapasitasVolumeSistem'] = dv_cleaned['KapasitasVolumeSistem'].fillna(dv_cleaned['KapasitasVolumeSistem'].median())

[ ] dv_cleaned['KapasitasDiskMemory'] = dv_cleaned['KapasitasDiskMemory'].fillna(dv_cleaned['KapasitasDiskMemory'].median())

[ ] dv_cleaned['TipeChassis'] = dv_cleaned['TipeChassis'].fillna(dv_cleaned['TipeChassis'].mode()[0])
```

```
cek_null(dv_cleaned)
```

```
Empty DataFrame
Columns: [Total, Percent]
Index: []
```

Encode (Platform)

```
[ ] dv_cleaned['Platform'].value_counts()
```

```
windows10      47820
windows7       44396
windows8       43051
windows2016    14338
Name: Platform, dtype: int64
```

```
from sklearn.preprocessing import OneHotEncoder

encoder = OneHotEncoder(sparse=False)
encoder = encoder.fit_transform(dv_cleaned[['Platform']])
dv_cleaned_platform = pd.DataFrame(encoder)
```

```
[ ] dv_cleaned_platform.value_counts()
```

```
0    1    2    3
1.0  0.0  0.0  0.0    47820
0.0  0.0  1.0  0.0    44396
               0.0  1.0    43051
          1.0  0.0  0.0    14338
dtype: int64
```

```
[ ] dv_cleaned_platform.rename(columns = {0:'Windows 10', 1:'Windows 7', 2:'Windows 8', 3:'Windows 2016'}, inplace = True)

dv_cleaned_platform
```

Encode (Platform)

```
dv_cleaned = dv_cleaned.join(dv_cleaned_platform)
dv_cleaned
```

	BanyakAntivirus	IdNegaraPembuat	IdKotaPembuat	IdLokasiGeografisMesinSaatIni	Platform	Processor	OsSuite	OsPlatformSubRelease	VersiInternetExplorer	Sm
0	1.0	68	151854.0	276.0	windows10	x64	768	th2	85.0	
1	1.0	57	117751.0	277.0	windows10	x64	768	prers5	163.0	
2	1.0	93	36825.0	119.0	windows10	x64	768	rs3	135.0	
3	1.0	50	115291.0	98.0	windows10	x86	768	rs2	108.0	
4	1.0	68	43129.0	150.0	windows10	x64	768	rs3	117.0	
...	...	...	...	...	...	...	...	...	...	
149663	1.0	29	11397.0	35.0	windows2016	x64	274	rs1	94.0	
149664	1.0	51	40629.0	98.0	windows2016	x64	400	rs1	96.0	
149665	1.0	51	13832.0	211.0	windows2016	x64	16	rs1	98.0	
149666	1.0	150	106860.0	192.0	windows2016	x64	272	rs1	98.0	
149667	1.0	97	89935.0	126.0	windows2016	x64	272	rs1	103.0	

149605 rows × 45 columns

```
[ ] dv_cleaned.drop('Platform', inplace=True, axis=1)
    dv_cleaned
```

Encode (Processor)

```
[ ]  encoder = OneHotEncoder(sparse=False)
     encoder = encoder.fit_transform(dv_cleaned[['Processor']])
     dv_cleaned_processor = pd.DataFrame(encoder)
```

```
[ ]  print(dv_cleaned_processor.value_counts())
     print(dv_cleaned['Processor'].value_counts())
```

```
0    1    2
0.0  1.0  0.0    129639
          0.0  1.0     19963
1.0  0.0  0.0         3
dtype: int64
x64      129639
x86       19963
arm64         3
Name: Processor, dtype: int64
```

```
     dv_cleaned_processor.rename(columns = {0:'x64', 1:'x86', 2:'arm64'}, inplace = True)
     dv_cleaned = dv_cleaned.join(dv_cleaned_processor)
     dv_cleaned.drop('Processor', inplace=True, axis=1)
     dv_cleaned
```

Encode (TipeBateraiInternal)

```
from sklearn.preprocessing import LabelEncoder

# TipeBateraiInternal
labelencoder = LabelEncoder()
# Assigning numerical values and storing in another column
dv_cleaned['TipeBateraiInternal_encode'] = labelencoder.fit_transform(dv_cleaned['TipeBateraiInternal'])
```

```
dv_cleaned.drop('TipeBateraiInternal', inplace=True, axis=1)
```

Encode (EdisiOS)

```
dv_cleaned['EdisiOS'].unique()
labelencoder = LabelEncoder()
# Assigning numerical values and storing in another column
dv_cleaned['EdisiOS_encode'] = labelencoder.fit_transform(dv_cleaned['EdisiOS'])
```

```
dv_cleaned.drop('EdisiOS', inplace=True, axis=1)
```

Encode (SkuNameOS)

```
dv_cleaned['SkuNameOS'].unique()
dv_cleaned['SkuNameOS_encode'] = labelencoder.fit_transform(dv_cleaned['SkuNameOS'])
```

```
dv_cleaned.drop('SkuNameOS', inplace=True, axis=1)
```

Encode (TipeInstalasiOS)

```
dv_cleaned['TipeInstallasiOS'].unique()
dv_cleaned['TipeInstallasiOS_encode'] = labelencoder.fit_transform(dv_cleaned['TipeInstallasiOS'])
```

```
dv_cleaned.drop('TipeInstallasiOS', inplace=True, axis=1)
```

Encode (AutoUpdateSetting)

```
dv_cleaned['AutoUpdateSetting'].unique()
dv_cleaned['AutoUpdateSetting_encode'] = labelencoder.fit_transform(dv_cleaned['AutoUpdateSetting'])
```

```
dv_cleaned.drop('AutoUpdateSetting', inplace=True, axis=1)
```

Encode (IsOSGenuine)

```
from sklearn.preprocessing import LabelEncoder

dv_cleaned['IsOSGenuine'] = dv_cleaned['IsOSGenuine'].replace(['OFFLINE', 'UNKNOWN'], 'INVALID_LICENSE')
print(dv_cleaned['IsOSGenuine'].value_counts())

# ISOSGenuine
labelencoder = LabelEncoder()
# Assigning numerical values and storing in another column
dv_cleaned['IsOSGenuine_encode'] = labelencoder.fit_transform(dv_cleaned['IsOSGenuine'])
print(dv_cleaned['IsOSGenuine'].value_counts())
print(dv_cleaned['IsOSGenuine_encode'].value_counts())
dv_cleaned.drop('IsOSGenuine', inplace=True, axis=1)
```

```
IS_GENUINE         134941
INVALID_LICENSE     14664
Name: IsOSGenuine, dtype: int64
IS_GENUINE         134941
INVALID_LICENSE     14664
Name: IsOSGenuine, dtype: int64
1    134941
0     14664
Name: IsOSGenuine_encode, dtype: int64
```

Encode (VersiOS)

```
dv_cleaned['VersiOS'].unique()
dv_cleaned['VersiOS_encode'] = labelencoder.fit_transform(dv_cleaned['VersiOS'])
```

```
dv_cleaned.drop('VersiOS', inplace=True, axis=1)
```

Encode (ArsitekturOS)

```
encoder = OneHotEncoder(sparse=False)
encoder = encoder.fit_transform(dv_cleaned[['ArsitekturOS']])
dv_cleaned_ArsitekturOS = pd.DataFrame(encoder)
```

```
dv_cleaned_ArsitekturOS.rename(columns = {0:'Arsitektur amd64', 1:'Arsitektur x86', 2:'Arsitektur arm64'}, inplace = True)
dv_cleaned = dv_cleaned.join(dv_cleaned_ArsitekturOS)
dv_cleaned.drop('ArsitekturOS', inplace=True, axis=1)
dv_cleaned
```

Encode (BranchOS)

```
dv_cleaned['BranchOS_encode'] = labelencoder.fit_transform(dv_cleaned['BranchOS'])
print(dv_cleaned['BranchOS'].value_counts())
print(dv_cleaned['BranchOS_encode'].value_counts())
```

```
dv_cleaned.drop('BranchOS', inplace=True, axis=1)
```

Encode (TipeChassis)

```
dv_cleaned['TipeChassis_encode'] = labelencoder.fit_transform(dv_cleaned['TipeChassis'])
print(dv_cleaned['TipeChassis'].value_counts())
print(dv_cleaned['TipeChassis_encode'].value_counts())
```

```
dv_cleaned.drop('TipeChassis', inplace=True, axis=1)
```

Encode (TipeDiskUtama)

```
encoder = OneHotEncoder(sparse=False)
encoder = encoder.fit_transform(dv_cleaned[['TipeDiskUtama']])
dv_cleaned_TipeDiskUtama = pd.DataFrame(encoder)
```

```
dv_cleaned_TipeDiskUtama.rename(columns = {0:'Disk HDD', 1:'Disk SSD', 2:'Disk UNKNOWN', 3:'Disk Unspecified'}, inplace = True)
dv_cleaned = dv_cleaned.join(dv_cleaned_TipeDiskUtama)
dv_cleaned.drop('TipeDiskUtama', inplace=True, axis=1)
dv_cleaned
```

Encode (SmartScreenSetting)

```
# SmartScreenSetting
dv_cleaned['SmartScreenSetting_encode'] = labelencoder.fit_transform(dv_cleaned['SmartScreenSetting'])
print(dv_cleaned['SmartScreenSetting'].value_counts())
print(dv_cleaned['SmartScreenSetting_encode'].value_counts())
```

```
dv_cleaned.drop('SmartScreenSetting', inplace=True, axis=1)
```

Encode (DeviceType)

```
# DeviceType
dv_cleaned['DeviceType_encode'] = labelencoder.fit_transform(dv_cleaned['DeviceType'])
print(dv_cleaned['DeviceType'].value_counts())
print(dv_cleaned['DeviceType_encode'].value_counts())
```

```
dv_cleaned.drop('DeviceType', inplace=True, axis=1)
```

Encode (OSPlatformSubrelease)

```
# OsPlatformSubRelease
dv_cleaned['OsPlatformSubRelease_encode'] = labelencoder.fit_transform(dv_cleaned['OsPlatformSubRelease'])
print(dv_cleaned['OsPlatformSubRelease'].value_counts())
print(dv_cleaned['OsPlatformSubRelease_encode'].value_counts())
```

```
windows7      44396
windows8.1    43051
rs4           21254
rs1           18438
rs3           14114
rs2            4402
th2            2289
th1            1532
prers5          129
Name: OsPlatformSubRelease, dtype: int64
7    44396
8    43051
4    21254
1    18438
3    14114
2     4402
6     2289
5     1532
0      129
Name: OsPlatformSubRelease_encode, dtype: int64
```

```
dv_cleaned.drop('OsPlatformSubRelease', inplace=True, axis=1)
```

Encode

```
dv_cleaned.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 149605 entries, 0 to 149667
Data columns (total 51 columns):
 #   Column                        Non-Null Count   Dtype
---  ------                        --------------   -----
 0   BanyakAntivirus               149605 non-null  float64
 1   IdNegaraPembuat               149605 non-null  int64
 2   IdKotaPembuat                 149605 non-null  float64
 3   IdLokasiGeografisMesinSaatIni 149605 non-null  float64
 4   OsSuite                       149605 non-null  int64
 5   VersiInternetExplorer         149605 non-null  float64
 6   IdOEM                         149605 non-null  float64
 7   IdModelOEM                    149605 non-null  float64
 8   BanyakCoreProcessor           149605 non-null  float64
 9   IdPembuatProcessor            149605 non-null  float64
 10  IdModelProcessor              149605 non-null  float64
 11  KapasitasDiskMemory           149605 non-null  float64
 12  KapasitasVolumeSistem         149605 non-null  float64
 13  KapasitasRAM                  149605 non-null  float64
 14  UkuranDiagonalLayar           149605 non-null  float64
 15  UkuranHorisontalLayar         149605 non-null  float64
 16  UkuranVertikalLayar           149605 non-null  float64
 17  BuildOS                       149605 non-null  int64
 18  RevisiBuildOS                 149605 non-null  int64
 19  IdPembuatFirmware             149605 non-null  float64
 20  IdVersiFirmware               149605 non-null  float64
 21  IsSecureBootEnabled           149605 non-null  int64
 22  IsTouchScreen                 149605 non-null  int64
 23  IsGamer                       149605 non-null  float64
 24  infected_proba                149605 non-null  float64
 25  Windows 10                    149544 non-null  float64
 26  Windows 7                     149544 non-null  float64
 27  Windows 8                     149544 non-null  float64
 28  Windows 2016                  149544 non-null  float64
 29  x64                           149544 non-null  float64
 30  x86                           149544 non-null  float64
 31  arm64                         149544 non-null  float64
 32  TipeBateraiInternal_encode    149605 non-null  int64
 33  EdisiOS_encode                149605 non-null  int64
 34  SkuNameOS_encode              149605 non-null  int64
 35  TipeInstallasiOS_encode       149605 non-null  int64
 36  AutoUpdateSetting_encode      149605 non-null  int64
 37  IsOSGenuine_encode            149605 non-null  int64
 38  VersiOS_encode                149605 non-null  int64
 39  Arsitektur amd64              149544 non-null  float64
 40  Arsitektur x86                149544 non-null  float64
 41  Arsitektur arm64              149544 non-null  float64
 42  BranchOS_encode               149605 non-null  int64
 43  TipeChassis_encode            149605 non-null  int64
 44  Disk HDD                      149544 non-null  float64
 45  Disk SSD                      149544 non-null  float64
 46  Disk UNKNOWN                  149544 non-null  float64
 47  Disk Unspecified              149544 non-null  float64
 48  SmartScreenSetting_encode     149605 non-null  int64
 49  DeviceType_encode             149605 non-null  int64
 50  OsPlatformSubRelease_encode   149605 non-null  int64
dtypes: float64(33), int64(18)
memory usage: 63.4 MB
```

Classification

04

Metrics Evaluasi Classification

```
[187] from sklearn.metrics import precision_score, \
          recall_score, classification_report, \
          accuracy_score, f1_score
      def evaluate_classifier_performance(prediction, y_test):
          # Informasi evaluasi secara compact
          print("Hasil Evaluasi berdasarkan classification report \n\n%s\n" % (classification_report(y_test, prediction,zero_division=0)))
          print()
          print("Confusion Matrix")
          print()
          y_actual = pd.Series(np.array(y_test), name = "actual")
          y_pred = pd.Series(np.array(prediction), name = "prediction")
          df_confusion = pd.crosstab(y_actual, y_pred)
          display(df_confusion)
          print()
          print()

          print("Butuh informasi lebih lengkap? silakan simak di bawah ini : ")
          print('F1 Macro Average:', f1_score(y_test, prediction, average='macro'))
          print('F1 Micro Average:', f1_score(y_test, prediction, average='micro'))
          print('Precision Macro Average:', precision_score(y_test, prediction, average='macro',zero_division=0))
          print('Precision Micro Average:', precision_score(y_test, prediction, average='micro',zero_division=0))
          print('Recall Macro Average:', recall_score(y_test, prediction, average='macro',zero_division=0))
          print('Recall Micro Average:', recall_score(y_test, prediction, average='micro',zero_division=0))
```

Persiapan Klasifikasi

```
# Memisahkan features and label
X_raw = dv_cleaned.drop('IsGamer', axis=1)
y = dv_cleaned['IsGamer']

selector = SelectKBest(f_classif, k=10)

X = selector.fit_transform(X_raw, y)
X.shape[1]
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/feature_selection/_univariate_selection.py:112: UserWarning: Features [0] are constant.
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
/usr/local/lib/python3.8/dist-packages/sklearn/feature_selection/_univariate_selection.py:113: RuntimeWarning: invalid value encountered in true_divide
  f = msb / msw
10
```

```
[382] input_features = selector.feature_names_in_
selector.get_feature_names_out(input_features=input_features)
```

```
array(['VersiInternetExplorer', 'UkuranHorisontalLayar', 'BuildOS',
       'Windows 10', 'Windows 8', 'Windows 2016',
       'TipeInstallasiOS_encode', 'VersiOS_encode', 'Disk SSD',
       'OsPlatformSubRelease_encode'], dtype=object)
```

Decision Tree Classification

```
[407] dt = DecisionTreeClassifier() # berdasarkan gridsearchcv di atas agar tidak run berkali2
      dt.fit(X_train_scaled,y_train)
      y_pred_dt = dt.predict(X_test_scaled)
```

```
evaluate_classifier_performance(y_pred_dt,y_test)
```

```
Hasil Evaluasi berdasarkan classification report

              precision    recall  f1-score   support

         0.0       0.91      0.97      0.94     33584
         1.0       0.44      0.20      0.27      3802

    accuracy                           0.89     37386
   macro avg       0.68      0.59      0.61     37386
weighted avg       0.87      0.89      0.87     37386
```

Confusion Matrix

prediction	0.0	1.0
actual		
0.0	32635	949
1.0	3048	754

```
Butuh informasi lebih lengkap? silakan simak di bawah ini :
F1 Macro Average: 0.6081142712425547
F1 Micro Average: 0.8930883218317017
Precision Macro Average: 0.6786646323553479
Precision Micro Average: 0.8930883218317017
Recall Macro Average: 0.585029585930426
Recall Micro Average: 0.8930883218317017
```

Random Forest Classification

```
[409] rf = RandomForestClassifier()
      rf.fit(X_train,y_train)
```

```
RandomForestClassifier()
```

```
[ ] y_pred_rf = rf.predict(X_test)
```

```
[ ] evaluate_classifier_performance(y_pred_rf,y_test)
```

```
Hasil Evaluasi berdasarkan classification report

              precision    recall  f1-score   support

         0.0       0.92      0.97      0.94     33584
         1.0       0.46      0.20      0.28      3802

    accuracy                           0.89     37386
   macro avg       0.69      0.59      0.61     37386
weighted avg       0.87      0.89      0.88     37386


Confusion Matrix
```

prediction	0.0	1.0
actual		
0.0	32672	912
1.0	3034	768

```
Butuh informasi lebih lengkap? silakan simak di bawah ini :
F1 Macro Average: 0.6116203285431864
F1 Micro Average: 0.8944524688386026
Precision Macro Average: 0.6860855718526698
Precision Micro Average: 0.8944524688386026
Recall Macro Average: 0.587421579725723
Recall Micro Average: 0.8944524688386026
```

Gaussian Naive Bayes Classification

```
[411] nb = GaussianNB()
      nb.fit(X_train,y_train)
```

```
      GaussianNB()
```

```
[412] y_pred_nb = nb.predict(X_test)
```

```
[413] evaluate_classifier_performance(y_pred_nb,y_test)
```

```
      Hasil Evaluasi berdasarkan classification report

                    precision    recall  f1-score   support

               0.0       0.98      0.77      0.86     33584
               1.0       0.30      0.87      0.45      3802

          accuracy                           0.78     37386
         macro avg       0.64      0.82      0.65     37386
      weighted avg       0.91      0.78      0.82     37386
```

Confusion Matrix

prediction	0.0	1.0
actual		
0.0	25911	7673
1.0	512	3290

```
Butuh informasi lebih lengkap? silakan simak di bawah ini :
F1 Macro Average: 0.6546238699062263
F1 Micro Average: 0.7810677793826566
Precision Macro Average: 0.6403616398163035
Precision Micro Average: 0.7810677793826566
Recall Macro Average: 0.8184310716708616
Recall Micro Average: 0.7810677793826566
```

KNN

```
evaluate_classifier_performance(y_pred_knn,y_test)
```

```
Hasil Evaluasi berdasarkan classification report

              precision    recall  f1-score   support

         0.0       0.91      0.98      0.94     33584
         1.0       0.48      0.14      0.22      3802

    accuracy                           0.90     37386
   macro avg       0.70      0.56      0.58     37386
weighted avg       0.87      0.90      0.87     37386


Confusion Matrix
```

prediction	0.0	1.0
actual		
0.0	32996	588
1.0	3253	549

```
Butuh informasi lebih lengkap? silakan simak di bawah ini :
F1 Macro Average: 0.5836547798859285
F1 Micro Average: 0.897261006793987
Precision Macro Average: 0.6965545987948644
Precision Micro Average: 0.897261006793987
Recall Macro Average: 0.5634446740626219
Recall Micro Average: 0.897261006793987
```

```
[615] scores = cross_val_score(KNN, X, y, scoring='accuracy', cv=cv)
      np.mean(scores)

0.8970002208070916
```

Logistic & Softmax Regression

```
evaluate_classifier_performance(y_logreg_predict,y_test)
```

```
Hasil Evaluasi berdasarkan classification report
```

	precision	recall	f1-score	support
0.0	0.90	0.99	0.94	33584
1.0	0.28	0.02	0.04	3802
accuracy			0.89	37386
macro avg	0.59	0.51	0.49	37386
weighted avg	0.84	0.89	0.85	37386

Confusion Matrix

prediction	0.0	1.0
actual		
0.0	33355	229
1.0	3711	91

```
Butuh informasi lebih lengkap? silakan simak di bawah ini :
F1 Macro Average: 0.49419272692434874
F1 Micro Average: 0.8946129567217675
Precision Macro Average: 0.5921281464145038
Precision Micro Average: 0.8946129567217675
Recall Macro Average: 0.5085580239857711
Recall Micro Average: 0.8946129567217675
```

```
[415] evaluate_classifier_performance(y_sofreg_predict,y_test)
```

```
Hasil Evaluasi berdasarkan classification report
```

	precision	recall	f1-score	support
0.0	0.90	0.99	0.94	33584
1.0	0.22	0.02	0.03	3802
accuracy			0.89	37386
macro avg	0.56	0.50	0.49	37386
weighted avg	0.83	0.89	0.85	37386

Confusion Matrix

prediction	0.0	1.0
actual		
0.0	33375	209
1.0	3743	59

```
Butuh informasi lebih lengkap? silakan simak di bawah ini :
F1 Macro Average: 0.48654802449037404
F1 Micro Average: 0.8942919809554378
Precision Macro Average: 0.5596543456005173
Precision Micro Average: 0.8942919809554378
Recall Macro Average: 0.504647473409221
Recall Micro Average: 0.8942919809554378
```

Analisis

Dapat dilihat dari 5 model klasifikasi yang dibuat, memiliki berbagai hasil evaluasi yang cukup mirip. Yaitu memiliki recall, precision, dan nilai F1 yang tinggi pada micro, namun kecil pada macro. Hal ini menandakan merupakan hal yang kurang baik, bisa juga menunjukkan bahwa dataset tidak balance. Hanya satu model yang berbeda, yaitu Gaussian Naive Bayes Classification. Memiliki recall, precision, dan nilai F1 yang tinggi pada micro namun tidak jauh berbeda dengan macronya. Sehingga model ini cocok digunakan untuk klasifikasi apakah mesin tersebut tergolong mesin gaming.

Regression

05

Metrics MAE,MSE,RMSE, R_SQUARED

```
# https://medium.com/analytics-vidhya/evaluation-metrics-for-regression-models-c91c65d73af
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
def metrics(prediction):
    MAE = mean_absolute_error(y_test, prediction)
    MSE = mean_squared_error(y_test, prediction)
    RMSE = np.sqrt(MSE)
    R_squared = r2_score(y_test, prediction)

    print('MAE: ' + str(MAE))
    print('MSE: ' + str(MSE))
    print('RMSE: ' + str(RMSE))
    print('R_squared: ' + str(R_squared))
```

Feature Selection

```
# Memisahkan features and label
X_raw = dv_cleaned.drop('infected_proba', axis=1)
y = dv_cleaned['infected_proba']

selector = SelectKBest(f_classif, k=25)

X = selector.fit_transform(X_raw, y)
X.shape[1]
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/feature_selection/_univaria
  warnings.warn("Features %s are constant." % constant_features_idx, UserW
/usr/local/lib/python3.8/dist-packages/sklearn/feature_selection/_univaria
  f = msb / msw
25
```

```
input_features = selector.feature_names_in_
selector.get_feature_names_out(input_features=input_features)
```

```
array(['IdLokasiGeografisMesinSaatIni', 'VersiInternetExplorer', 'IdOEM',
       'IdModelOEM', 'BanyakCoreProcessor', 'KapasitasDiskMemory',
       'KapasitasVolumeSistem', 'KapasitasRAM', 'UkuranDiagonalLayar',
       'UkuranHorisontalLayar', 'RevisiBuildOS', 'IsGamer', 'Windows 7',
       'Windows 8', 'Windows 2016', 'EdisiOS_encode',
       'TipeInstallasiOS_encode', 'AutoUpdateSetting_encode',
       'BranchOS_encode', 'TipeChassis_encode', 'Disk HDD',
       'Disk UNKNOWN', 'SmartScreenSetting_encode', 'DeviceType_encode',
       'OsPlatformSubRelease_encode'], dtype=object)
```

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif
```

Melakukan feature selection, untuk membantu dalam memilih fitur

Split & Normalization Data

```
# Memisahkan features and label
X = dv_cleaned[['VersiInternetExplorer','BanyakCoreProcessor', 'KapasitasDiskMemory',
       'KapasitasVolumeSistem', 'KapasitasRAM', 'UkuranDiagonalLayar',
       'UkuranHorisontalLayar', 'RevisiBuildOS', 'IsGamer', 'Windows 7',
       'Windows 8', 'Windows 2016', 'EdisiOS_encode',
       'TipeInstallasiOS_encode', 'AutoUpdateSetting_encode',
       'BranchOS_encode', 'TipeChassis_encode', 'Disk HDD',
       'Disk UNKNOWN', 'SmartScreenSetting_encode', 'DeviceType_encode',
       'OsPlatformSubRelease_encode']]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```

```
from sklearn.preprocessing import StandardScaler

train_feature = X_train

scaler = StandardScaler()
scaled_data = scaler.fit_transform(train_feature)
#normalization fitur pada dataset training
X_train = pd.DataFrame(scaled_data, columns=train_feature.columns)

test_feature = X_test

scaler = StandardScaler()
scaled_data = scaler.fit_transform(test_feature)
#normalization fitur pada dataset training
X_test = pd.DataFrame(scaled_data, columns=test_feature.columns)
```

Random Forest

```
from sklearn.ensemble import RandomForestRegressor

# Melakukan training pada model random forest
rf_dv = RandomForestRegressor()
rf_dv.fit(X_train, y_train)

# Memprediksi data testing
predicted = rf_dv.predict(X_test)

# Menampilkan metrics
metrics(predicted)
```

```
MAE: 0.3071196623887776
MSE: 0.12667054671045083
RMSE: 0.3559080593502356
R_squared: 0.007316688372480562
```

```
from sklearn.metrics import accuracy_score

rf_dv.score(X_test,y_test)
```

```
0.007316688372480562
```

Linear Regression

```
# Melatih model linear regression menggunakan Scikit-learn pada training set

from sklearn.linear_model import LinearRegression

linear = LinearRegression()
linear.fit(X_train, y_train)
```

```
LinearRegression()
```

```
linear_test = LinearRegression()
linear_test.fit(X_test, y_test)
y_predict = linear_test.predict(X_test)

# Menampilkan metrics
metrics(y_predict)
```

```
MAE: 0.30411287491664885
MSE: 0.12120300463074245
RMSE: 0.34814221897199205
R_squared: 0.050164358324945924
```

```
linear.score(X_test,y_test)
```

```
0.04961805171691258
```

Lasso

```
from sklearn.linear_model import Lasso

alpha = [1, 5, 10, 15, 20, 25]

# 3b. Bangun (fit) model sebanyak nilai parameter alpha yang dipilih

# List nilai R-squared
list_r_squared_train = []
list_r_squared_test = []
lasreg = Lasso()

# Membangun dan mengevaluasi model untuk setiap nilai alpha
for a in alpha:

  # Membangun model
  lasreg.set_params(**{'alpha': a})
  lasreg.fit(X_train, y_train)

  # Mengevaluasi model pada training dan testing set menggunakan R-squared
  r_squared_train = lasreg.score(X_train, y_train)
  r_squared_test = lasreg.score(X_test, y_test)

  # Menambahkan nilai R-squared ke list
  list_r_squared_train.append(r_squared_train)
  list_r_squared_test.append(r_squared_test)
```

```
print("R-squared training:", list_r_squared_train)
print("R-squared testing:", list_r_squared_test)
```

```
R-squared training: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
R-squared testing: [-1.129720823467295e-05, -1.129720823467295e-05, -1.129720823467295e-05, -1.129720823467295e-05, -1.129720823467295e-05, -1.129720823467295e-05]
```

```
lasreg.score(X_test,y_test)
```

```
-1.129720823467295e-05
```

Decision Tree

```
[235] tree_model2 = DecisionTreeRegressor(criterion="squared_error", max_depth=5, min_samples_split = 20, random_state = 2023)
      tree_model2.fit(X_train, y_train)
      predicted = tree_model2.predict(X_test)
      metrics(predicted)

      MAE: 0.3029788017018353
      MSE: 0.12043650705318398
      RMSE: 0.3470396332599261
      R_squared: 0.056171195536949714

      tree_model2.score(X_test,y_test)

      0.056171195536949714
```

Ridge

```
[320] # Melatih model ridge regression menggunakan Scikit-learn pada training set

      from sklearn.linear_model import Ridge

      ridge = Ridge(alpha=5) # Mengatur hyperparameter alpha=5
      ridge.fit(X_train, y_train)
```

```
Ridge(alpha=5)
```

```
[322] y_predict = ridge.predict(X_test)

      print(y_predict)
```

```
[0.449655   0.452872   0.43645452 ... 0.47553679 0.44592157 0.39837987]
```

```
# Menampilkan metrics
metrics(y_predict)
```

```
MAE: 0.3044263653314261
MSE: 0.12127261870543905
RMSE: 0.348242183983272
R_squared: 0.049618811376580196
```

Analisis

Dari lima model yang telah diujikan, kelimanya memiliki kemiripan, yaitu pada nilai MAE, MSE, dan RMSE. Semua nilai R-squared dari tiap model yang didapat nilainya kecil, namun MAE, MSE, dan RMSE juga kecil. Dengan perbandingan tersebut, menurut kami model yang cocok dipilih adalah **Decision Tree Regressor.** Karena memiliki nilai MAE, MSE, dan RMSE terkecil dibandingkan model lainnya dan nilai R-squared nya yang paling besar. Sehingga model ini cocok untuk memprediksi apakah mesin tersebut mungkin terinfeksi virus.

Clustering

06

```
[ ] X_clust4 = dv_clust[['BanyakCoreProcessor', 'KapasitasRAM']]
    X_clust4
```

	BanyakCoreProcessor	KapasitasRAM
0	2.0	2048.0
1	4.0	6144.0
2	4.0	4096.0
3	2.0	2048.0
4	4.0	4096.0
...	...	...
149600	7.0	14336.0
149601	2.0	4096.0
149602	4.0	4096.0
149603	2.0	10240.0
149604	7.0	14336.0

149544 rows × 2 columns

K-Means

```
[ ] from sklearn.cluster import KMeans
    from sklearn.metrics import silhouette_samples, silhouette_score
    from yellowbrick.cluster import SilhouetteVisualizer

    fig, ax = plt.subplots(3, 2, figsize=(20,10))
    for k in [2, 3, 4, 5, 6]:
        # Create KMeans instance for different number of clusters
        clusterer = KMeans(n_clusters = k)

        # Draw silhouette diagram
        q, mod = divmod(k, 2)
        visualizer = SilhouetteVisualizer(clusterer, colors = 'yellowbrick', ax = ax[q-1][mod])
        visualizer.fit(X_clust4)

        # Compute silhoutte score
        # This gives a perspective into the density and separation of the formed clusters
        cluster_labels = clusterer.fit_predict(X_clust4)
        silhouette_avg = silhouette_score(X_clust4, cluster_labels)
        print(
            "For n_clusters =",
            k,
            "The average silhouette_coefficient is :",
            silhouette_avg,
        )
```

```
For n_clusters = 2 The average silhouette_coefficient is : 0.7539259983300611
For n_clusters = 3 The average silhouette_coefficient is : 0.8201444652066455
For n_clusters = 4 The average silhouette_coefficient is : 0.8746233734595859
For n_clusters = 5 The average silhouette_coefficient is : 0.9381070720854658
For n_clusters = 6 The average silhouette_coefficient is : 0.9500427728879085
```

K-Means

```
[ ] kmeans = KMeans(n_clusters=6)
    cluster_assignment = kmeans.fit_predict(X_clust4)
    data_with_clusters = pd.DataFrame(X_clust4.copy(), columns=('BanyakCoreProcessor', 'KapasitasRAM'))
    data_with_clusters['Clusters'] = cluster_assignment
    data_with_clusters
```

	BanyakCoreProcessor	KapasitasRAM	Clusters
0	2.0	2048.0	3
1	4.0	6144.0	4
2	4.0	4096.0	0
3	2.0	2048.0	3
4	4.0	4096.0	0
...	...	...	...
149600	7.0	14336.0	2
149601	2.0	4096.0	0
149602	4.0	4096.0	0
149603	2.0	10240.0	5
149604	7.0	14336.0	2

149544 rows × 3 columns

K-Means

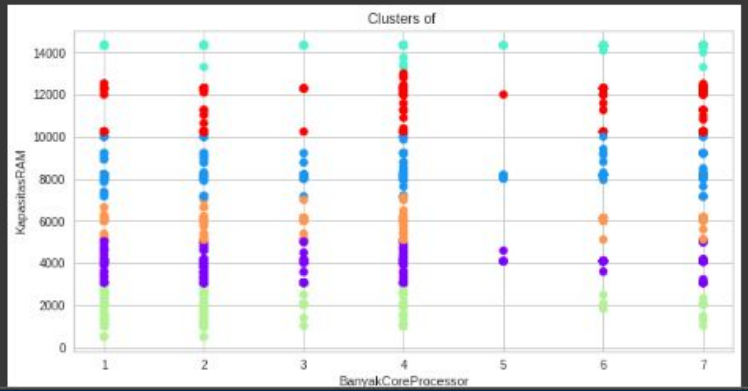
```
[ ]  # Create figure
     fig = plt.figure(figsize = (10, 5))
     ax = plt.axes()

     # Prepare data
     x = data_with_clusters['BanyakCoreProcessor']
     y = data_with_clusters['KapasitasRAM']
     cluster = data_with_clusters['Clusters']

     # Create plot
     ax.scatter(x, y, c = cluster, cmap = "rainbow")
     plt.title("Clusters of ")
     ax.set_xlabel('BanyakCoreProcessor')
     ax.set_ylabel('KapasitasRAM')

     # Show plot
     plt.show()

     print(y)
```



Analisis

Dari pengelompokan yang telah dilakukan, kelompok kami memilih atribut Kapasitas RAM dan banyak core processor karena secara intuitif atribut tersebut dapat dimanfaatkan untuk mengelompokkan mesin berdasarkan cara kerjanya. Pengelompokan akan memiliki nilai silhouette coefficient yang maksimal jika menggunakan n=6 untuk K-Nearest Neighbor.

Kesimpulan

07

Model Terbaik

Dari hasil percobaan yang sudah dilakukan, kami menentukan bahwa model yang terbaik yang dapat digunakan untuk melakukan classification, regression, dan clustering adalah sebagai berikut:

- Classification : Gaussian Naive Bayes Classification
- Regression : Decision Tree
- Clustering : K-Means

THANKS

CREDITS: This presentation template was created by Slidesgo,
including icons by Flaticon, and infographics & images by Freepik