Klasifikasi Ketersediaan Akses Internet Warga Desa Jawa Barat

Ika Lulus Yuliatin

Keterangan Project

Saat ini kita memasuki revolusi industri generasi keempat, Revolusi industri generasi keempat menjadi tantangan sekaligus menjadi peluang bagi sumber daya manusia di bidang teknologi informasi dan komunikasi (SDM TIK) di Indonesia.

Dikutip dari laman resmi KOMINFO, Menurut Menteri Johnny, Kementerian Kominfo berupaya melakukan percepatan pembangunan infrastruktur digital dari hulu hingga hilir yang dimulai dari lapisan backbone, middle mile, hingga the last mile. Dengan percepatan pembangunan jaringan Teknologi Informasi dan Komunikasi itu, pemerintah berharap pemanfaatan teknologi digital oleh publik akan meningkat dan menjangkau seluruh pelosok tanah air.

Pada Analisis ini akan dilakukan visualisasi data mengenai sumber daya dan teknologi masyarakat Desa di Jawa Barat dan juga Klasifikasi desa yang sudah memiliki akses internet dan tidak dengan menggunakan 10 Model yang dibagi menjadi 75% data training dan 25% data testing.

10 model Klasifikasi Mechine Learning yang digunakan adalah

1. Suport Vector Classifier
2. KNN
3. Decision Tree Classifier
4. Random Forest Classifier
5. Ada Boost Classifier
6. Gradient Boosting Classifier
7. Stochastic Gradient Boosting (SGB)
8. XgBoost Classifier
9. Extra Trees Classifier
10. LGBM Classifier

Data pada analisis ini terdiri dari 6 variabel numerik dan 9 variable kategorik. Data yang digunakan pada analisis bersumber dari website Open Data Jabar yang dikeluarkan oleh Dinas Pemberdayaan Masyarakat dan Desa Pemerintah Provinsi Jawa Barat.

Model Performance Analysis

1. Confusion Matrix

Confusion Matrix adalah teknik yang digunakan untuk meringkas kinerja algoritma klasifikasi yaitu memiliki keluaran biner.

n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Dalam contoh ketersedian internet pada suatu desa Peristiwa ini memperkirakan YA (suatu desa memiliki Ketersediaan akses internet warga), dan memang benar desa tersebut memiliki Ketersediaan akses internet warga disebut sebagai TRUE POSITIF (TP).

Kasus di mana diperkirakan TIDAK (desa tidak memiliki Ketersediaan akses internet warga), dan memang benar desa tersebut tidak terdapat akses internet disebut sebagai NEGATIF BENAR (TN).

Kasus di mana memperkirakan YA (suatu desa memiliki Ketersediaan akses internet warga), dan mereka tidak memiliki Ketersediaan akses internet warga akan disebut sebagai FALSE POSITIF (FP). Juga dikenal sebagai “Kesalahan tipe I”

Kasus di mana diperkirakan TIDAK (desa tidak memiliki Ketersediaan akses internet warga), dan ternyata suatu desa memiliki Ketersediaan akses internet warga akan disebut sebagai FALSE NEGATIVE (FN). Juga dikenal sebagai "Kesalahan Tipe II".

2. Classification Report

Report akan terdiri dari Precision, Recall and F1-Score.

Precision Score

```
TP – True Positives
FP – False Positives

Precision – Accuracy of positive predictions.
Precision = TP/(TP + FP)
```

Recall Score

```
FN – False Negatives

Recall(sensitivity or true positive rate): Fraction of positives that were correctly identified.
Recall = TP/(TP+FN)
```

F1 Score

```
F1 Score (aka F-Score or F-Measure) – A helpful metric for comparing two classifiers.
F1 Score takes into account precision and the recall.
```

It is created by finding the the harmonic mean of precision and recall.

F1 = 2 x (precision x recall)/(precision + recall)

Accuracy Score

Accuracy Score = (TP+TN)/ (TP+FN+TN+FP)

Library yang Dibutuhkan

```
#  imports yang dibutuhkan

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import colors
import seaborn as sns
import plotly.express as px
```

```
from google.colab import files
data_to_load = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving SDM TIK new.csv to SDM TIK new.csv

```
df = pd.read_csv('SDM TIK new.csv')
```

```
df.head()
```

	Kabupaten/Kota	Kecamatan	Kelurahan	Keberadaan_papan_informasi	Keberadaan_web
0	BOGOR	GUNUNG PUTRI	WANAHERANG	ADA	
1	BOGOR	GUNUNG PUTRI	BOJONG KULUR	ADA	
2	BOGOR	GUNUNG PUTRI	CIANGSANA	ADA	
3	BOGOR	GUNUNG PUTRI	GUNUNG PUTRI	ADA	
4	BOGOR	GUNUNG PUTRI	BOJONG NANGKA	ADA	

```
# kita dapat melihat beberapa nilai yang hilang dilambangkan dengan '?' jadi mari kita ganti nilai yang hil

df.replace('?', np.nan, inplace = True)
```

```
df.describe()
```

	Frekuensi_musyawarah desa	Jumlah_bahasa	Jumlah_pasar_semi_permanen	Jumlah_alat_tek
count	5312.000000	5312.000000	5312.000000	
mean	6.762425	1.878012	2.932229	
std	7.158772	0.910094	32.847346	
min	0.000000	0.000000	0.000000	
25%	3.000000	1.000000	0.000000	
50%	5.000000	2.000000	0.000000	
75%	8.000000	2.000000	0.000000	
max	180.000000	25.000000	1500.000000	

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5312 entries, 0 to 5311
Data columns (total 18 columns):
 #   Column                                        Non-Null Count  Dtype 
---  ------                                        --------------  ----- 
 0   Kabupaten/Kota                                5312 non-null   object
 1   Kecamatan                                     5312 non-null   object
 2   Kelurahan                                     5312 non-null   object
 3   Keberadaan_papan_informasi                    5312 non-null   object
 4   Keberadaan_website_desa                       5312 non-null   object
 5   Keberadaan_sarana_informasi_lainnya           5312 non-null   object
 6   Frekuensi_musyawarah desa                     5312 non-null   int64 
 7   Jumlah_bahasa                                 5312 non-null   int64 
 8   Jumlah_pasar_semi_permanen                    5312 non-null   int64 
 9   Jumlah_alat_teknologi_tepat_guna_peternakan   5312 non-null   int64 
 10  Jumlah_alat_teknologi_tepat_guna_pertanian    5312 non-null   int64 
 11  Jumlah_alat_teknologi_tepat_guna_perikanan    5312 non-null   int64 
 12  Ketersediaan_sinyal_telkomsel                 5312 non-null   object
 13  Ketersediaan_sinyal_indosat                   5312 non-null   object
 14  Ketersediaan_sinyal_xl                        5312 non-null   object
 15  Ketersediaan_sinyal_operator_lainnya          5312 non-null   object
 16  Status_sinyal_telepon_seluler                 5312 non-null   object
 17  Ketersediaan_akses_internet_warga             5312 non-null   object
dtypes: int64(6), object(12)
memory usage: 747.1+ KB
```

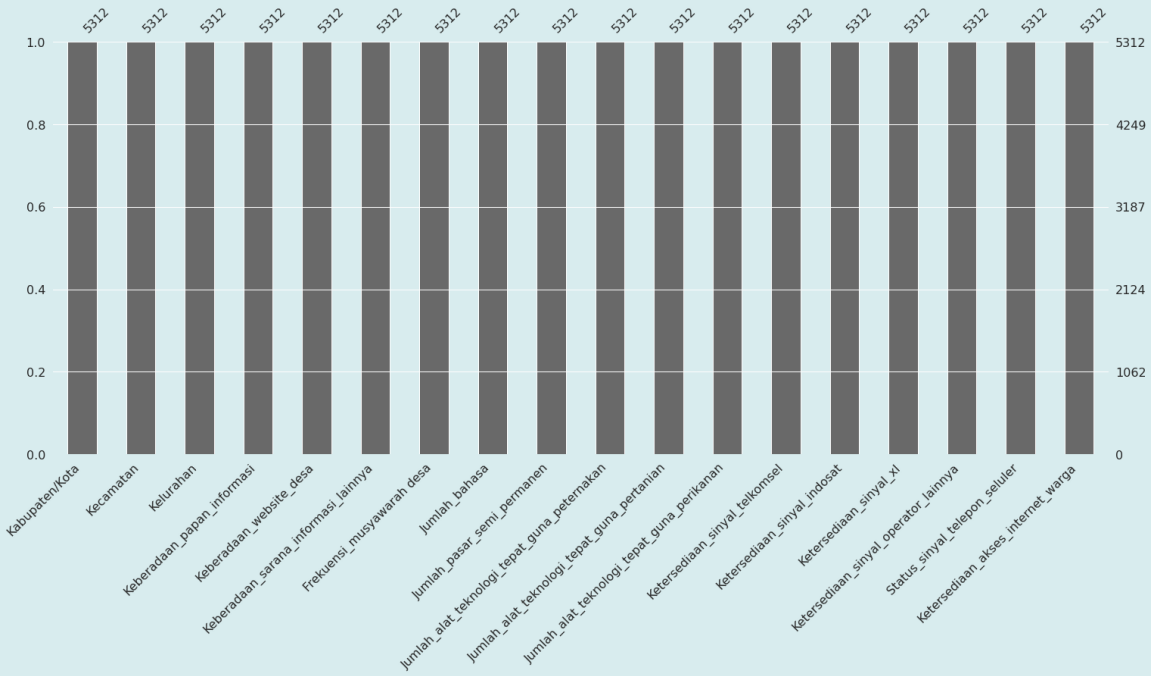
Data Pre-Processing

```
# missing values
df.isna().sum()
```

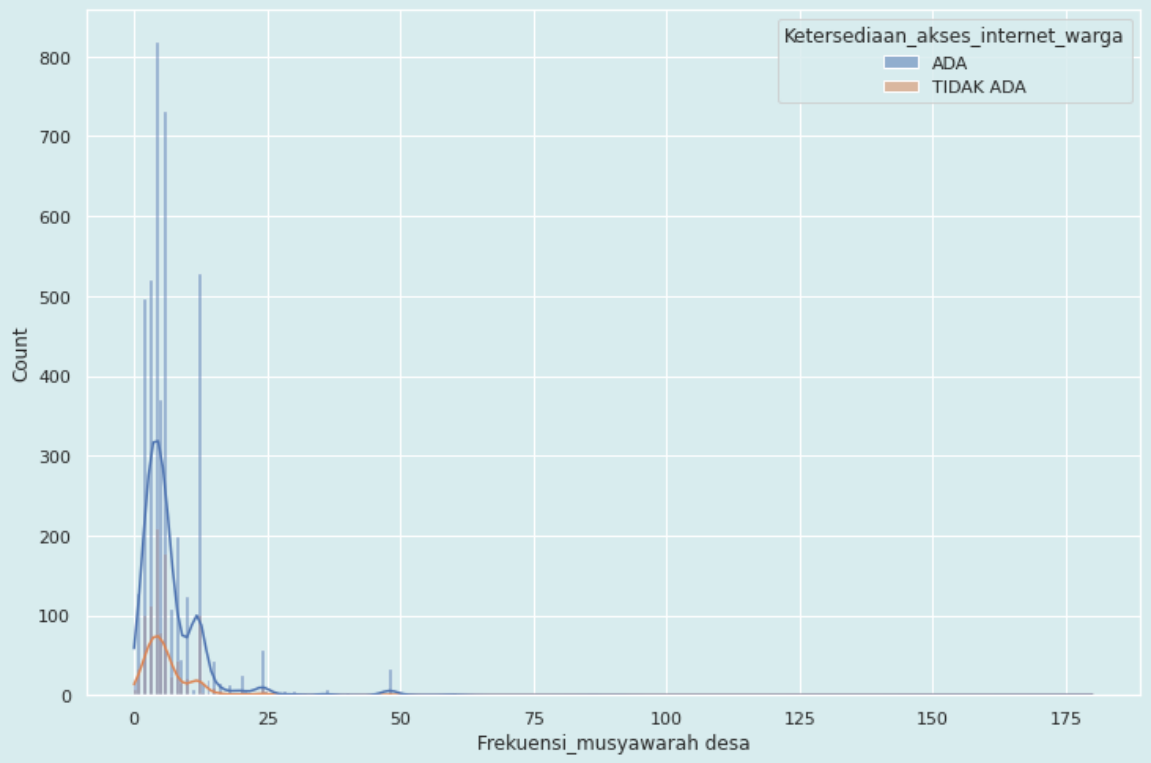
```
Kabupaten/Kota                                0
Kecamatan                                     0
Kelurahan                                     0
Keberadaan_papan_informasi                    0
Keberadaan_website_desa                       0
Keberadaan_sarana_informasi_lainnya           0
```

```
Frekuensi_musyawarah desa                    0
Jumlah_bahasa                                0
Jumlah_pasar_semi_permanen                   0
Jumlah_alat_teknologi_tepat_guna_peternakan  0
Jumlah_alat_teknologi_tepat_guna_pertanian   0
Jumlah_alat_teknologi_tepat_guna_perikanan   0
Ketersediaan_sinyal_telkomsel                0
Ketersediaan_sinyal_indosat                  0
Ketersediaan_sinyal_xl                       0
Ketersediaan_sinyal_operator_lainnya         0
Status_sinyal_telepon_seluler                0
Ketersediaan_akses_internet_warga            0
dtype: int64
```

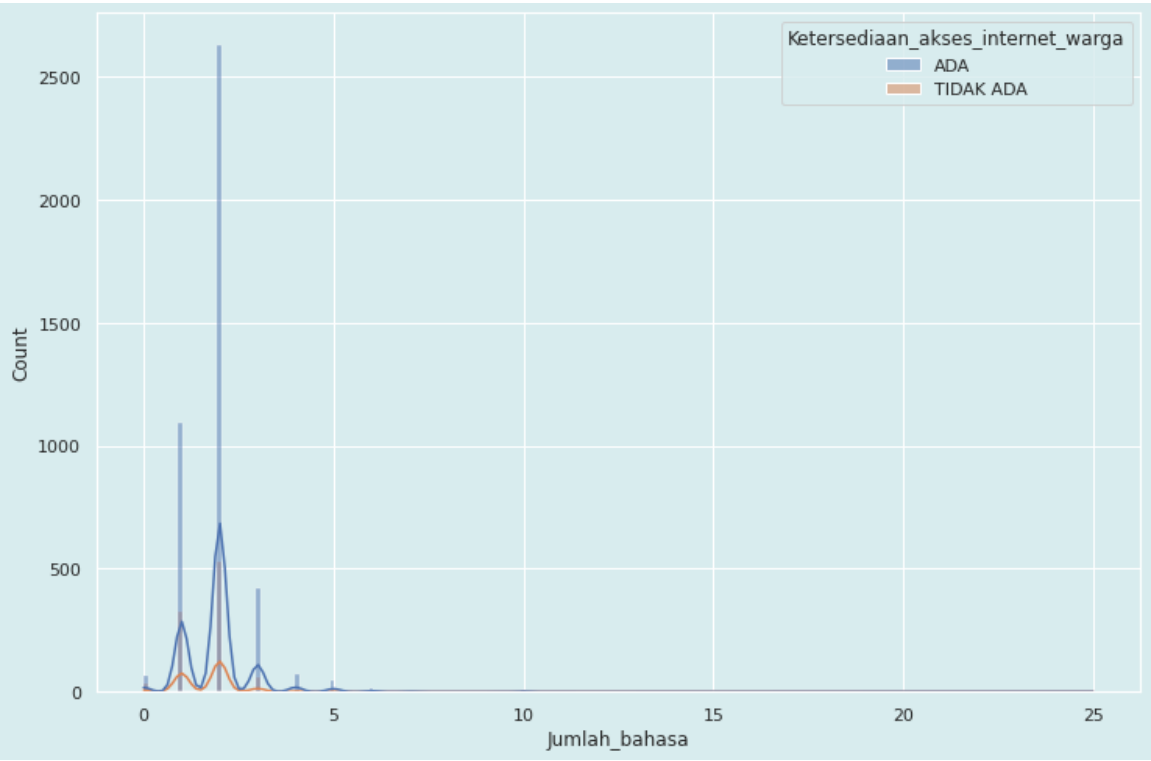
```
import missingno as msno
msno.bar(df)
plt.show()
```



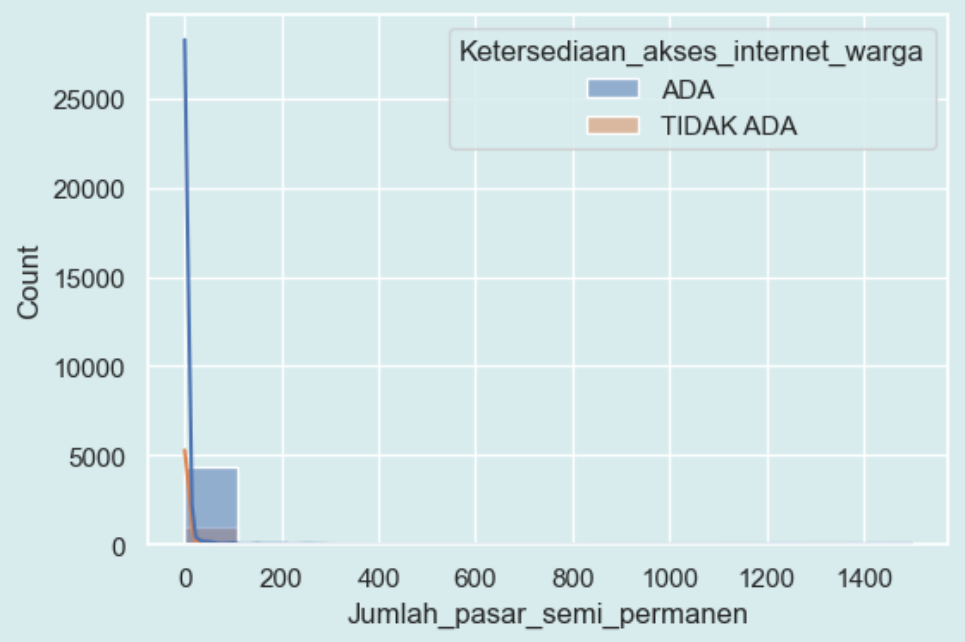

```
sns.set(rc={"axes.facecolor":"#d8ecee","figure.facecolor":"#d8ecee"})
pallet = ["#682F2F", "#9E726F", "#D6B2B1", "#B9C0C9", "#9F8A78", "#6f33f0"]
cmap = colors.ListedColormap(["#682F2F", "#9E726F", "#D6B2B1", "#B9C0C9", "#9F8A78", "#6f33f0"])
plt.figure(figsize=(12,8))
sns.histplot(data=df,x='Frekuensi_musyawarah desa',kde=True,hue='Ketersediaan_akses_internet_warga');
```



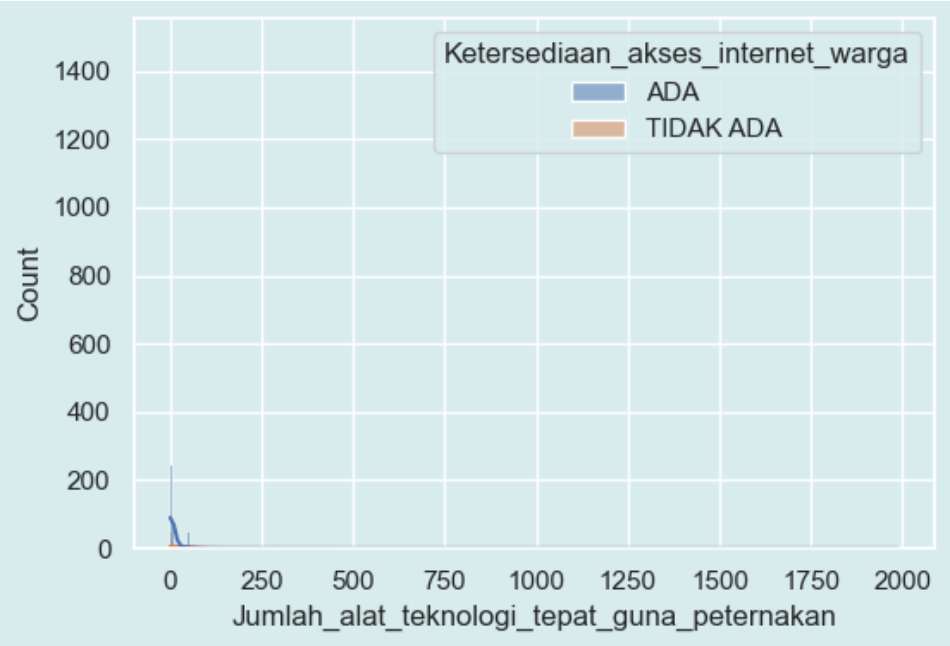

```
plt.figure(figsize=(12,8))
sns.histplot(data=df,x='Jumlah_bahasa',kde=True,hue='Ketersediaan_akses_internet_warga');
```



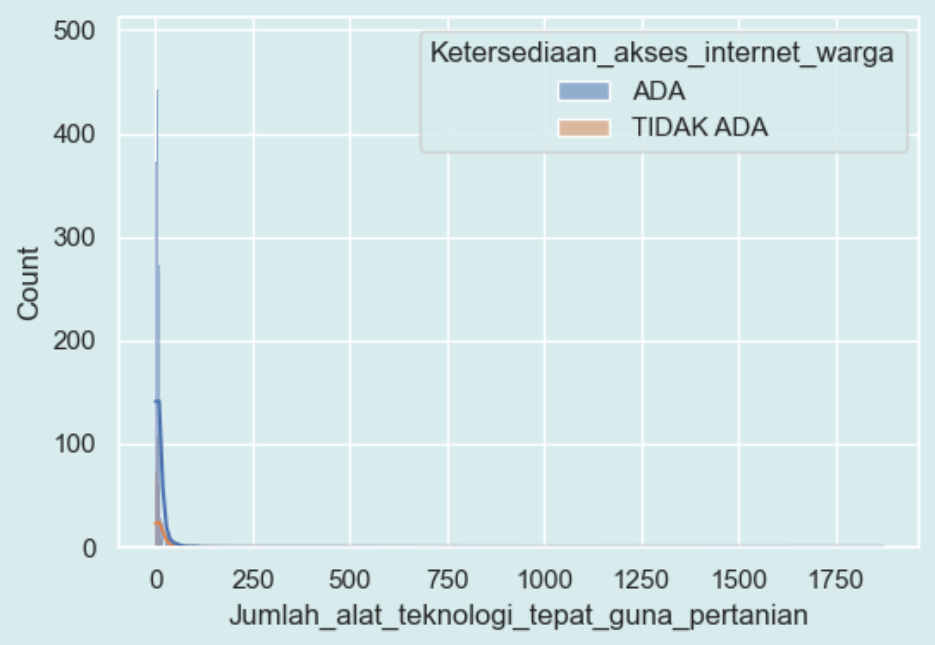
```
plt.figure(figsize=(6,4))
sns.histplot(data=df,x='Jumlah_pasar_semi_permanen',kde=True,hue='Ketersediaan_akses_internet_warga');
```



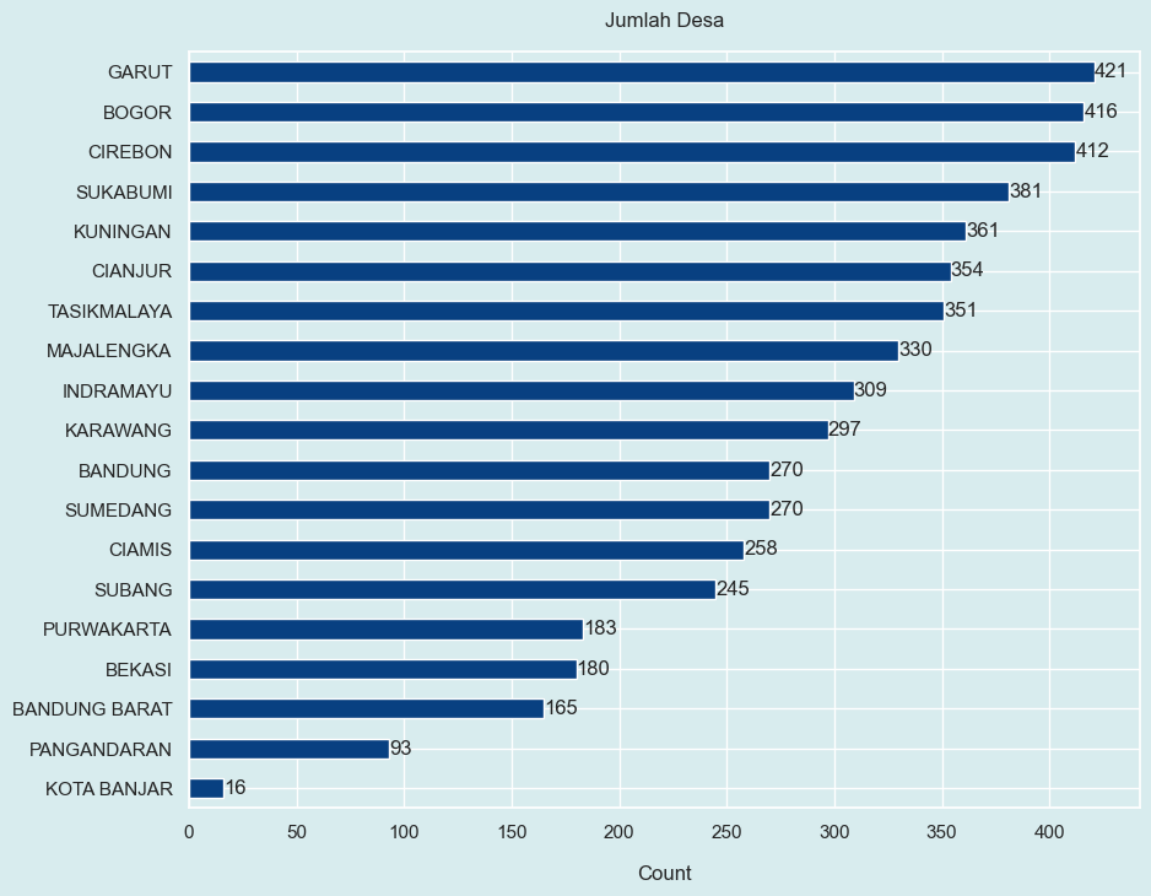

```
plt.figure(figsize=(6,4))
sns.histplot(data=df,x='Jumlah_alat_teknologi_tepat_guna_peternakan',kde=True,hue='Ketersediaan_akses_inter
```



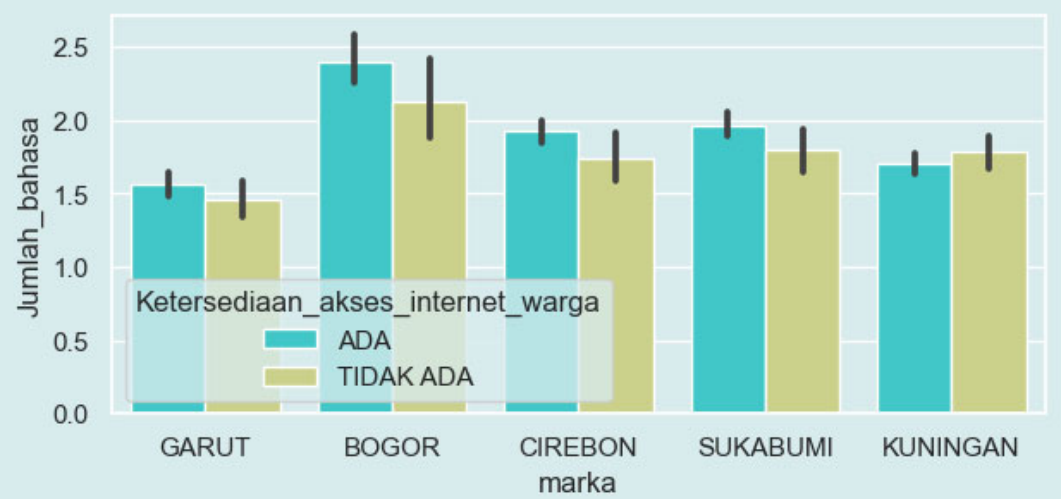

```
plt.figure(figsize=(6,4))
sns.histplot(data=df,x='Jumlah_alat_teknologi_tepat_guna_pertanian',kde=True,hue='Ketersediaan_akses_intern
```

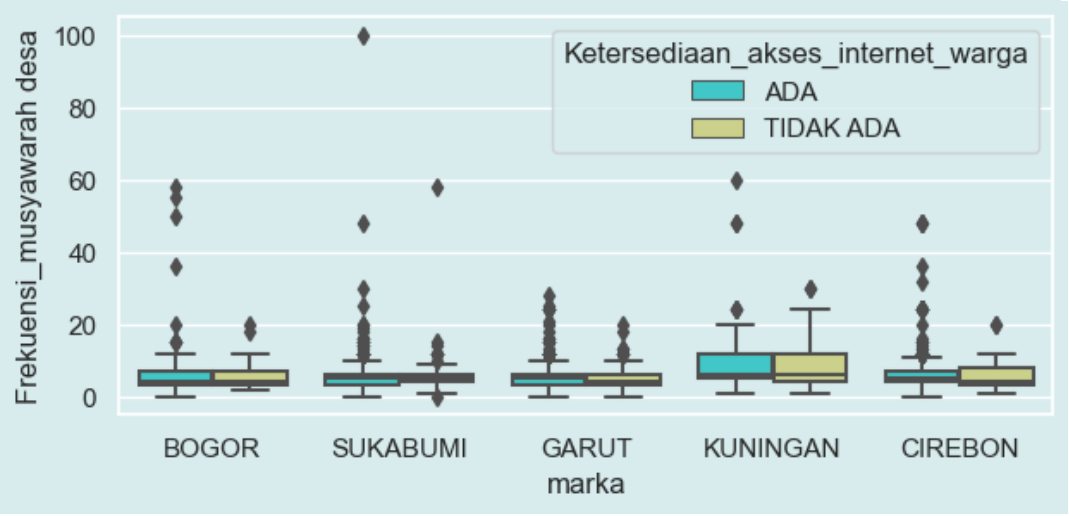



```
sns.set(rc={"axes.facecolor":"#d8ecee","figure.facecolor":"#d8ecee"})
pallet = ["#682F2F", "#9E726F", "#D6B2B1", "#B9C0C9", "#9F8A78", "#6f33f0"]
cmap = colors.ListedColormap(["#682F2F", "#9E726F", "#D6B2B1", "#B9C0C9", "#9F8A78", "#6f33f0"])

dx = df['Kabupaten/Kota'].value_counts().head(19).sort_values().plot(kind='barh', figsize=(10,8), cmap='GnB
dx.bar_label(dx.containers[0], )
plt.xlabel("Count", labelpad=14)
plt.title("Jumlah Desa", y=1.02);
```

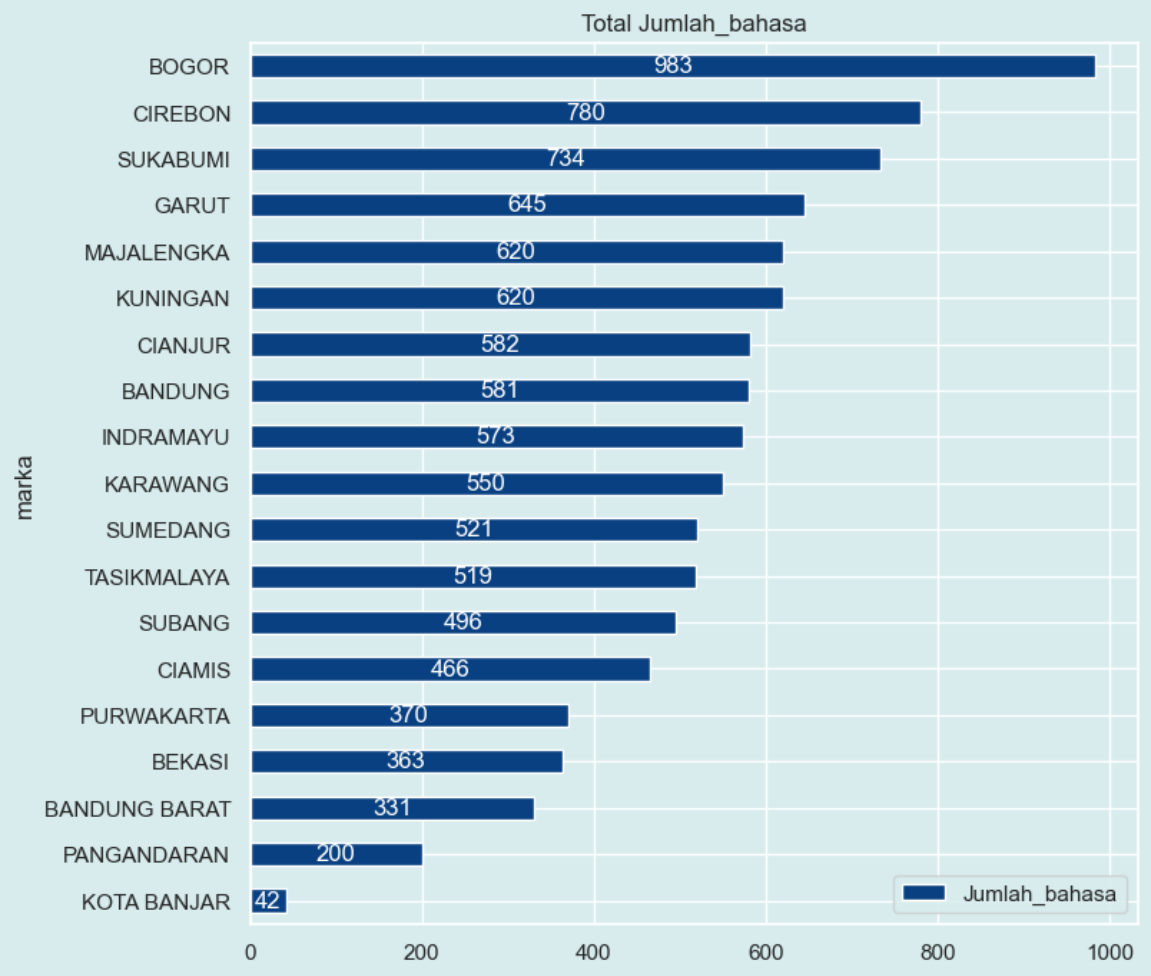



```
df['marka'] = df['Kabupaten/Kota']


big_5 = ["GARUT", "BOGOR", "CIREBON", "SUKABUMI", "KUNINGAN"]
big_5_df = df[df.marka.isin(big_5)]
fig, ax = plt.subplots(figsize = (7,3))
sns.barplot(data = big_5_df, x = "marka", y = "Jumlah_bahasa", order = big_5, hue ="Ketersediaan_akses_inte
plt.show

fig, ax = plt.subplots(figsize = (7,3))
sns.boxplot(data = df[df.marka.isin(big_5)], x = "marka", y = "Frekuensi_musyawarah desa", hue ="Ketersedia
plt.show()
```

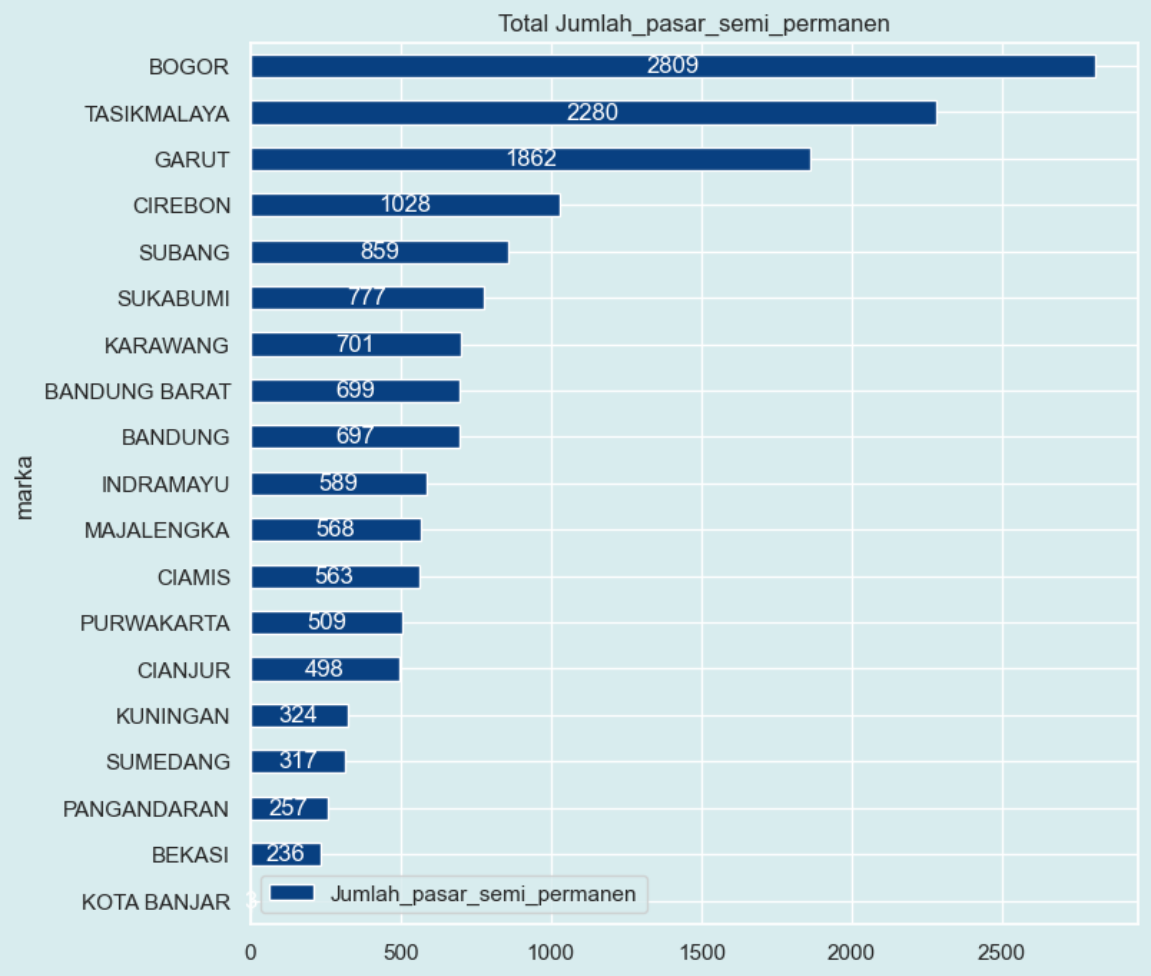




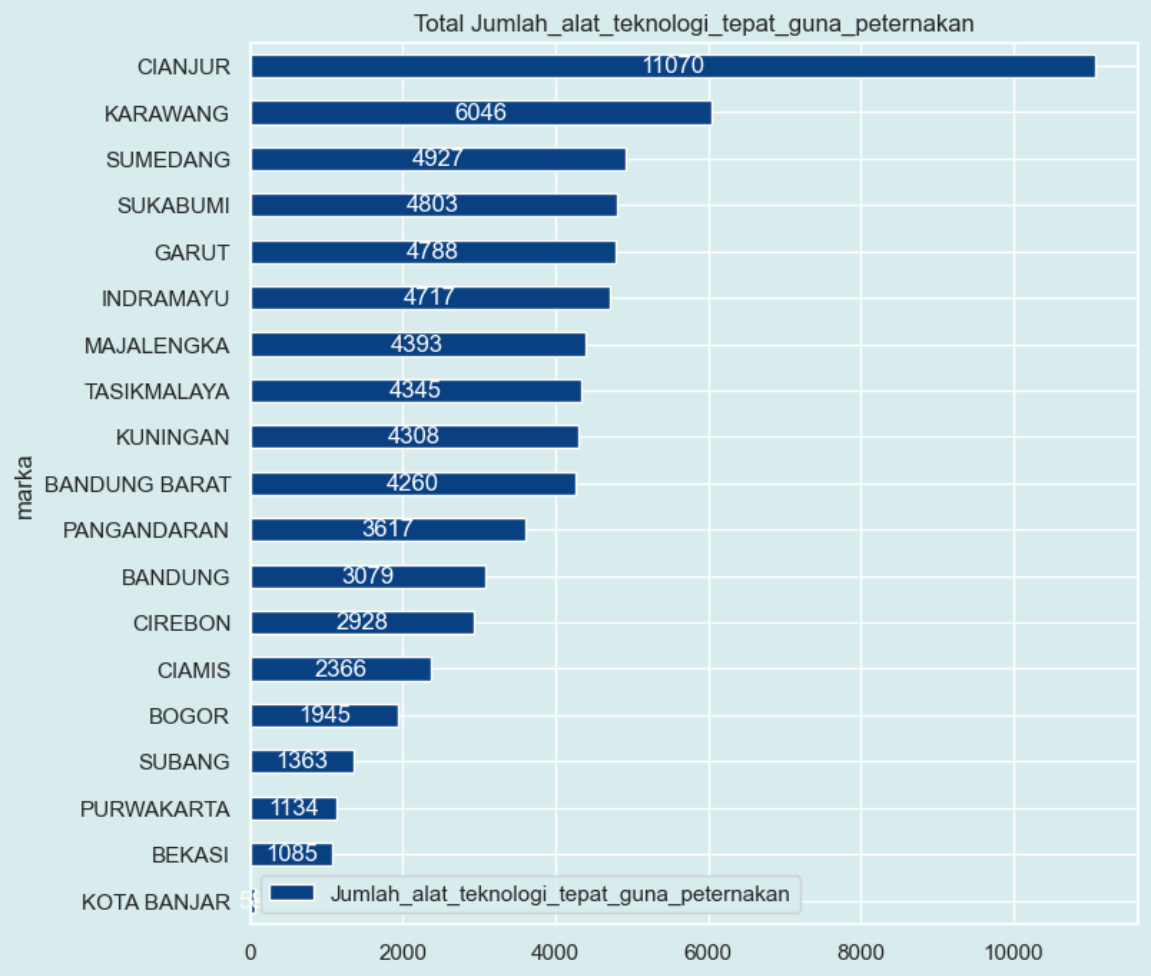

```
Jumlah_bahasa = df.groupby(['marka'])[['Jumlah_bahasa']].sum()
Jumlah_bahasa.sort_values(by='Jumlah_bahasa', ascending=True, inplace=True)
ax   = Jumlah_bahasa.plot(kind='barh', cmap='GnBu_r' , figsize=(8,8) ,title= 'Total Jumlah_bahasa', )
for c in ax.containers:
        # set the bar label
    ax.bar_label(c, fmt='%.0f',label_type='center', color='w',rotation=0)
```



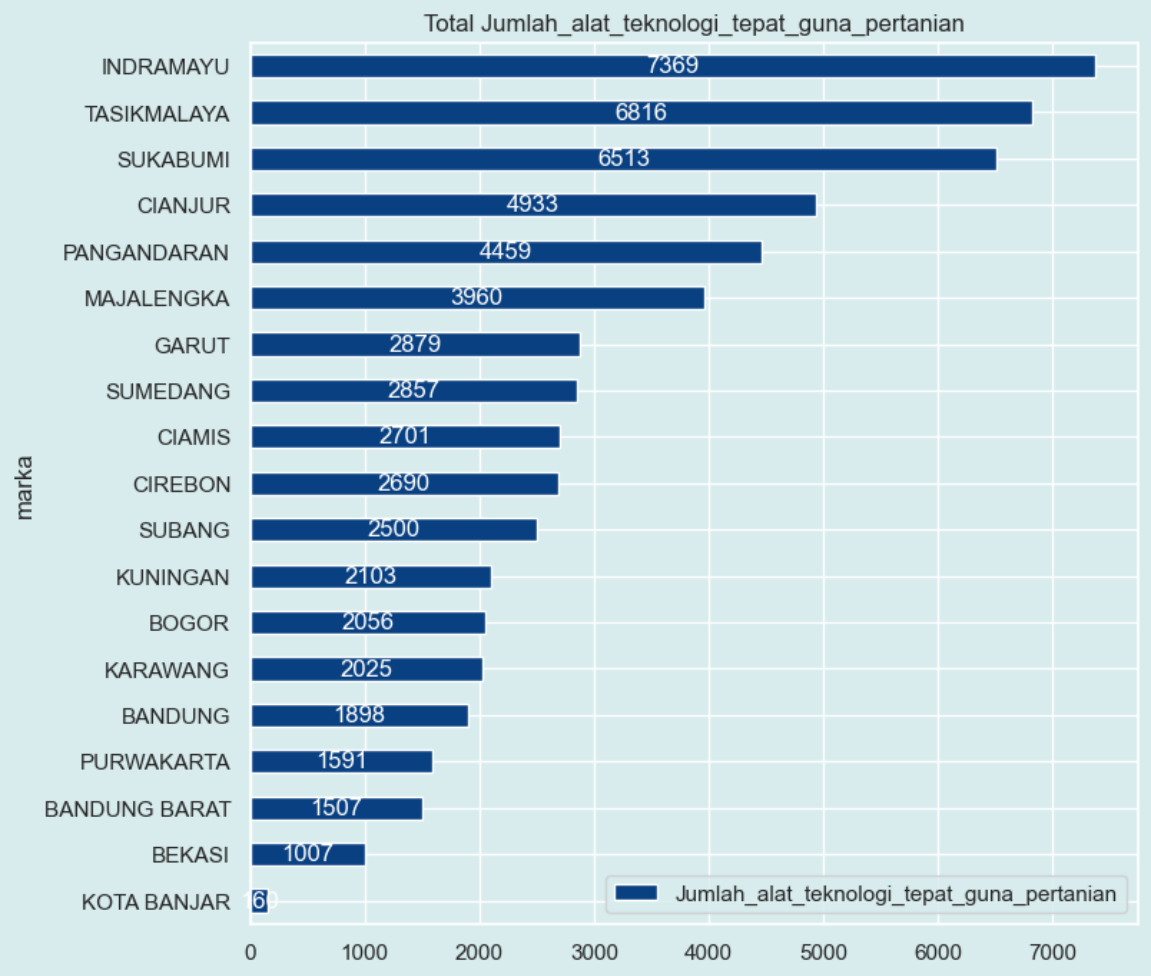

```
Jumlah_pasar_semi_permanen = df.groupby(['marka'])[['Jumlah_pasar_semi_permanen']].sum()
Jumlah_pasar_semi_permanen.sort_values(by='Jumlah_pasar_semi_permanen', ascending=True, inplace=True)
ax   = Jumlah_pasar_semi_permanen.plot(kind='barh', cmap='GnBu_r' , figsize=(8,8) ,title= 'Total Jumlah_pas
for c in ax.containers:
        # set the bar label
    ax.bar_label(c, fmt='%.0f',label_type='center', color='w',rotation=0)
```



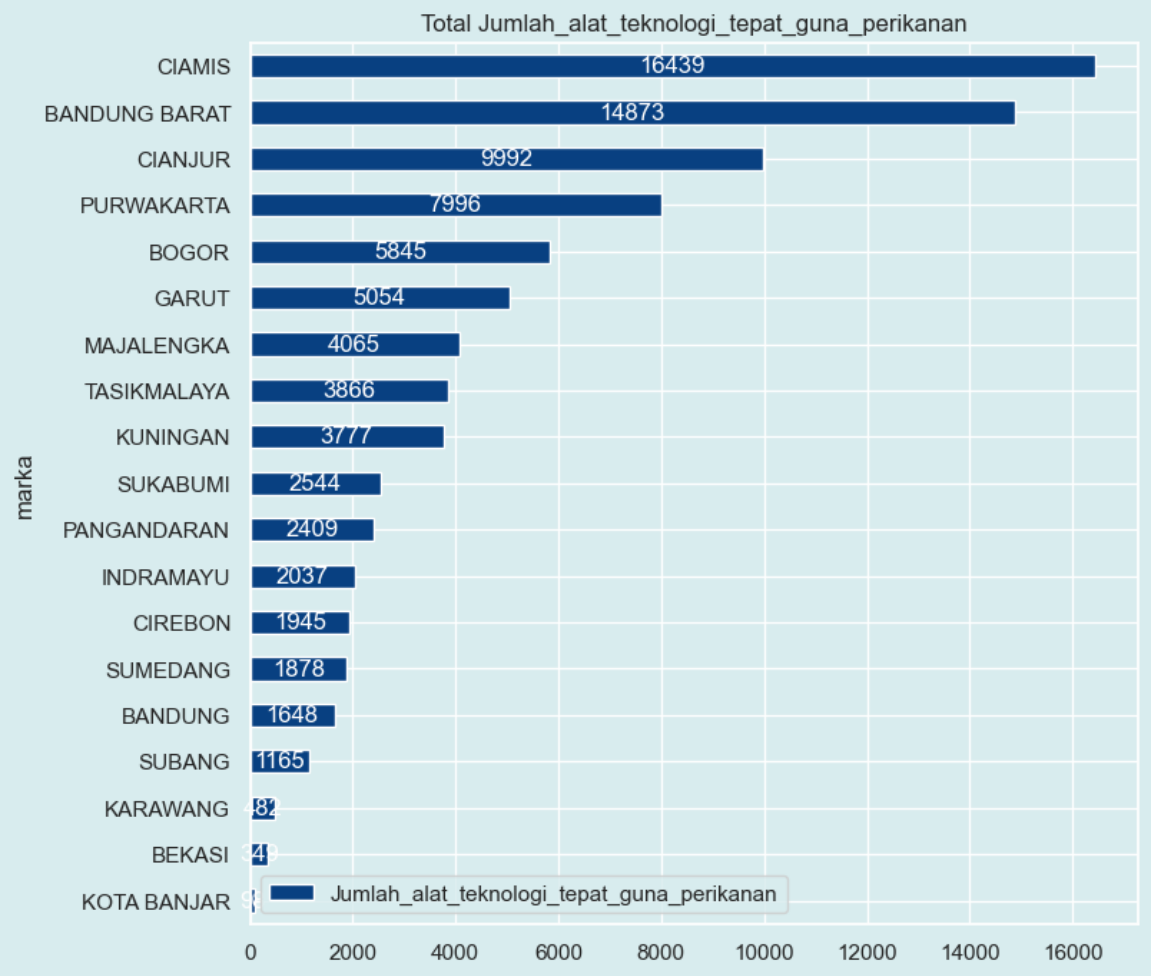

```
Jumlah_alat_teknologi_tepat_guna_peternakan = df.groupby(['marka'])[['Jumlah_alat_teknologi_tepat_guna_pete
Jumlah_alat_teknologi_tepat_guna_peternakan.sort_values(by='Jumlah_alat_teknologi_tepat_guna_peternakan', a
ax   = Jumlah_alat_teknologi_tepat_guna_peternakan.plot(kind='barh', cmap='GnBu_r' , figsize=(8,8) ,title=
for c in ax.containers:
        # set the bar label
    ax.bar_label(c, fmt='%.0f',label_type='center', color='w',rotation=0)
```



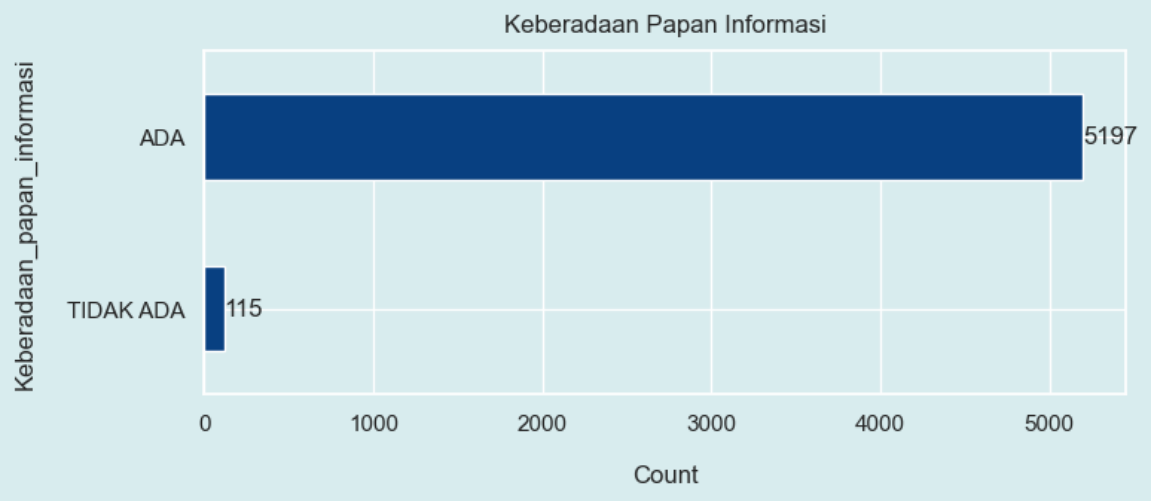

```
Jumlah_alat_teknologi_tepat_guna_pertanian = df.groupby(['marka'])[['Jumlah_alat_teknologi_tepat_guna_perta
Jumlah_alat_teknologi_tepat_guna_pertanian.sort_values(by='Jumlah_alat_teknologi_tepat_guna_pertanian', asc
ax   = Jumlah_alat_teknologi_tepat_guna_pertanian.plot(kind='barh', cmap='GnBu_r' , figsize=(8,8) ,title= '
for c in ax.containers:
        # set the bar label
    ax.bar_label(c, fmt='%.0f',label_type='center', color='w',rotation=0)
```




```
Jumlah_alat_teknologi_tepat_guna_perikanan = df.groupby(['marka'])[['Jumlah_alat_teknologi_tepat_guna_perik
Jumlah_alat_teknologi_tepat_guna_perikanan.sort_values(by='Jumlah_alat_teknologi_tepat_guna_perikanan', asc
ax   = Jumlah_alat_teknologi_tepat_guna_perikanan.plot(kind='barh', cmap='GnBu_r' , figsize=(8,8) ,title= '
for c in ax.containers:
        # set the bar label
    ax.bar_label(c, fmt='%.0f',label_type='center', color='w',rotation=0)
```

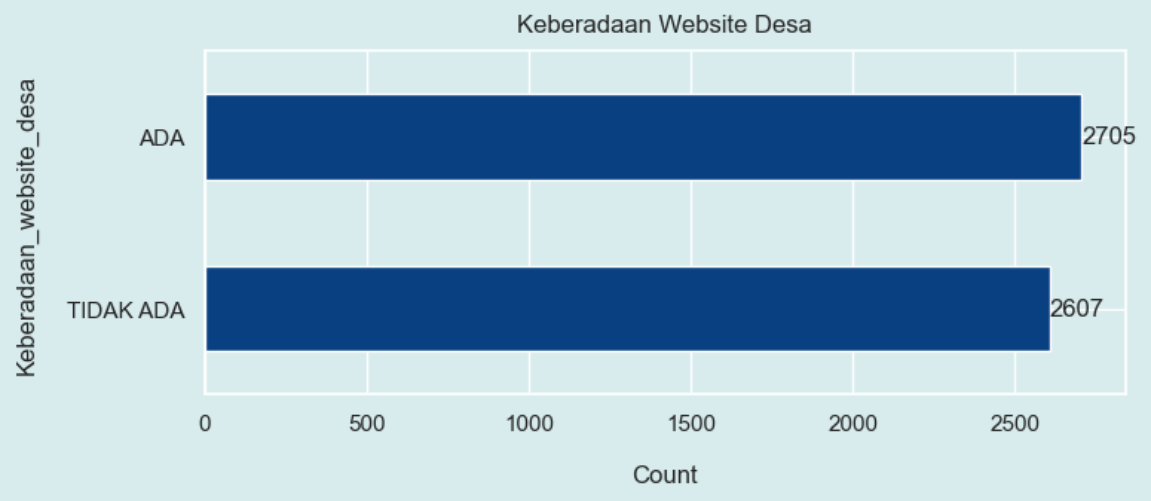



```
dx = df['Keberadaan_papan_informasi'].value_counts().head().sort_values().plot(kind='barh', figsize=(8,3),
dx.bar_label(dx.containers[0], )
plt.xlabel("Count", labelpad=14)
plt.ylabel("Keberadaan_papan_informasi", labelpad=14)
plt.title("Keberadaan Papan Informasi", y=1.02);
```

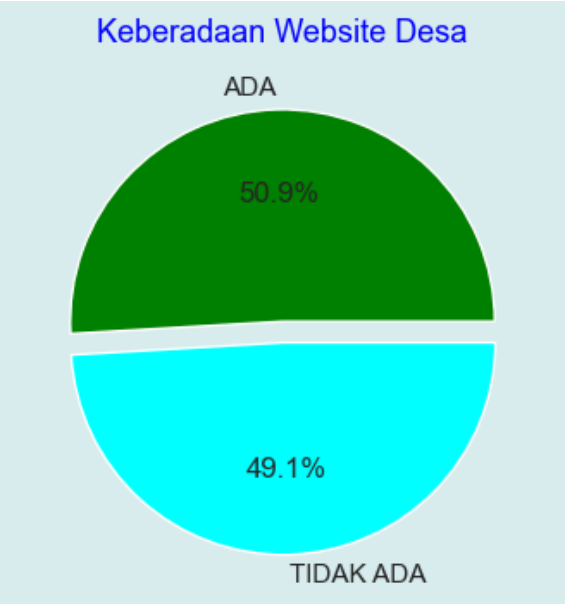



```
labels=df['Keberadaan_papan_informasi'].value_counts().index
colors=['green','cyan']
explode=[0,0.1]
values=df['Keberadaan_papan_informasi'].value_counts().values

#visualization
plt.figure(figsize=(4,4))
plt.pie(values,explode=explode,labels=labels,colors=colors,autopct='%1.1f%%')
plt.title('Keberadaan Papan Informasi',color='Blue',fontsize=13)
plt.show()
```

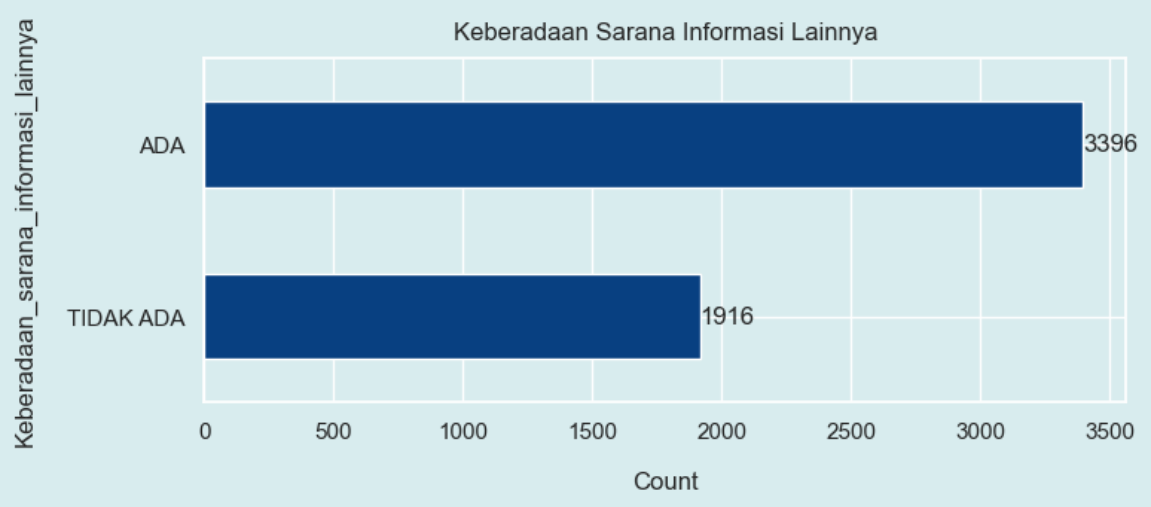

```
dx = df['Keberadaan_website_desa'].value_counts().head().sort_values().plot(kind='barh', figsize=(8,3), cma
dx.bar_label(dx.containers[0], )
plt.xlabel("Count", labelpad=14)
plt.ylabel("Keberadaan_website_desa", labelpad=14)
plt.title("Keberadaan Website Desa", y=1.02);
```



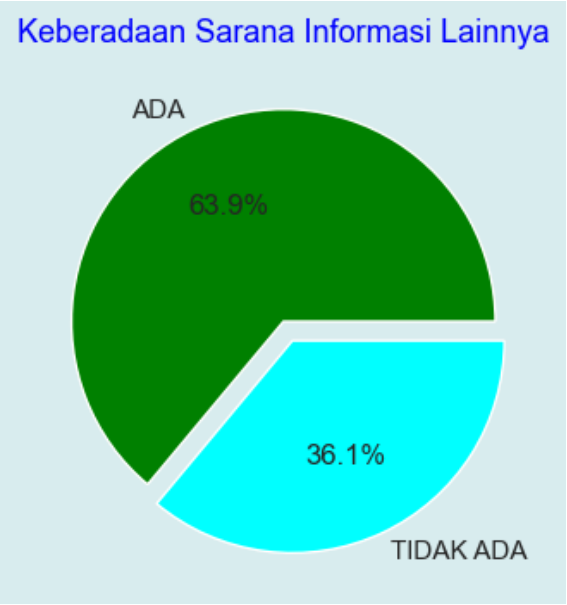

```
labels=df['Keberadaan_website_desa'].value_counts().index
colors=['green','cyan']
explode=[0,0.1]
values=df['Keberadaan_website_desa'].value_counts().values

#visualization
plt.figure(figsize=(4,4))
plt.pie(values,explode=explode,labels=labels,colors=colors,autopct='%1.1f%%')
plt.title('Keberadaan Website Desa',color='Blue',fontsize=13)
plt.show()
```



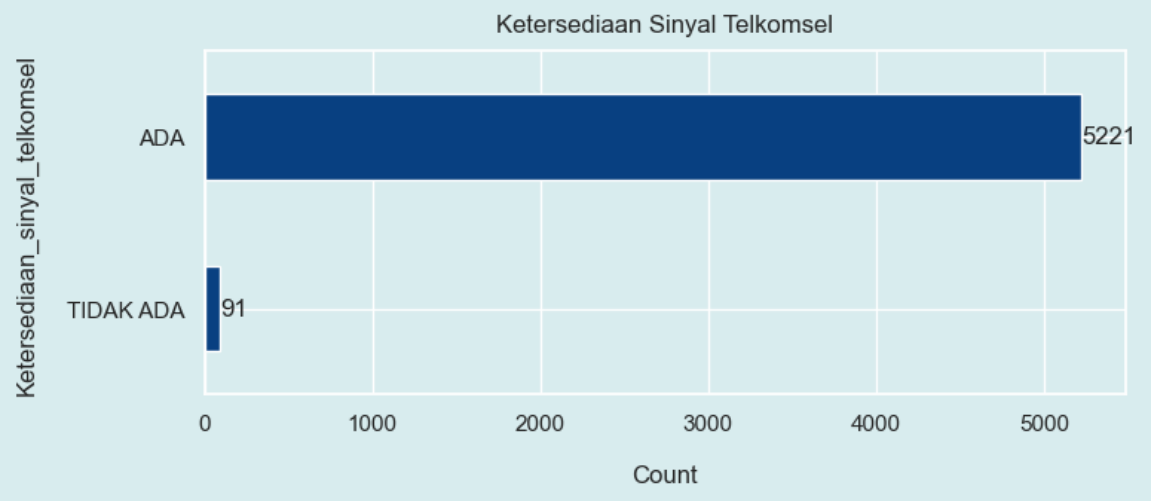

```
dx = df['Keberadaan_sarana_informasi_lainnya'].value_counts().head().sort_values().plot(kind='barh', figsiz
dx.bar_label(dx.containers[0], )
plt.xlabel("Count", labelpad=14)
plt.ylabel("Keberadaan_sarana_informasi_lainnya", labelpad=14)
plt.title("Keberadaan Sarana Informasi Lainnya", y=1.02);
```



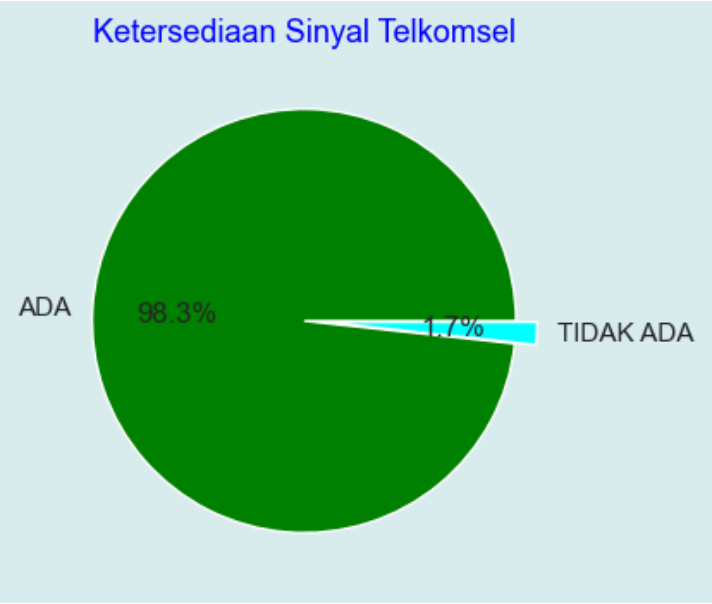

```
labels=df['Keberadaan_sarana_informasi_lainnya'].value_counts().index
colors=['green','cyan']
explode=[0,0.1]
values=df['Keberadaan_sarana_informasi_lainnya'].value_counts().values

#visualization
plt.figure(figsize=(4,4))
plt.pie(values,explode=explode,labels=labels,colors=colors,autopct='%1.1f%%')
plt.title('Keberadaan Sarana Informasi Lainnya',color='Blue',fontsize=13)
plt.show()
```



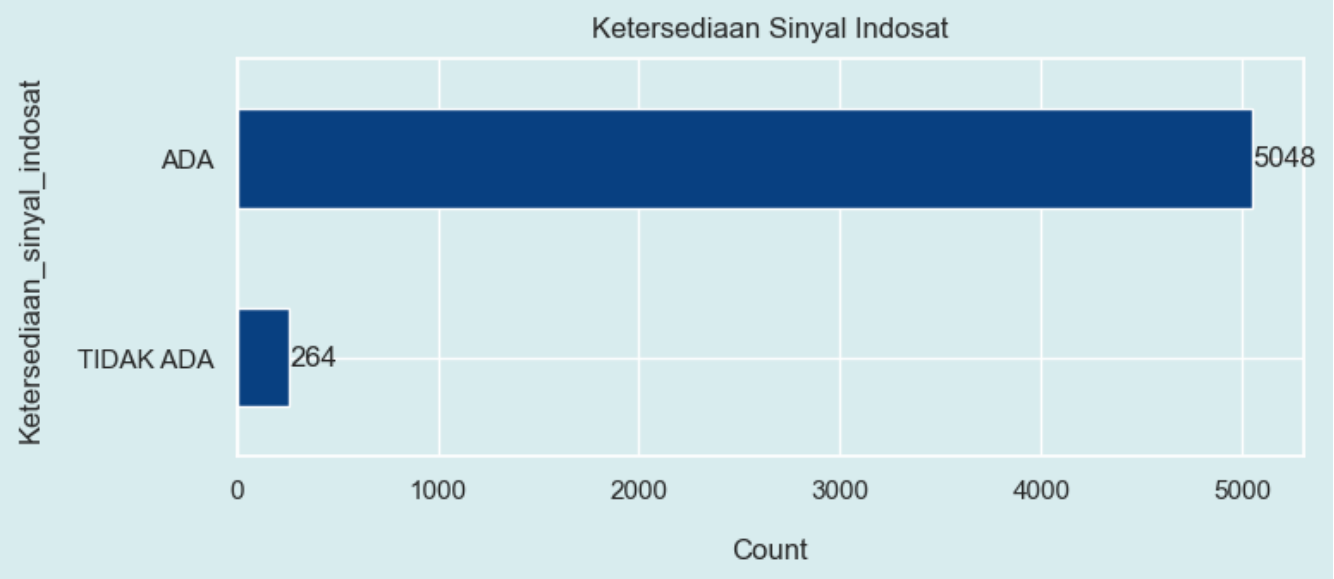

```
dx = df['Ketersediaan_sinyal_telkomsel'].value_counts().head().sort_values().plot(kind='barh', figsize=(8,3
dx.bar_label(dx.containers[0], )
plt.xlabel("Count", labelpad=14)
plt.ylabel("Ketersediaan_sinyal_telkomsel", labelpad=14)
plt.title("Ketersediaan Sinyal Telkomsel", y=1.02);
```



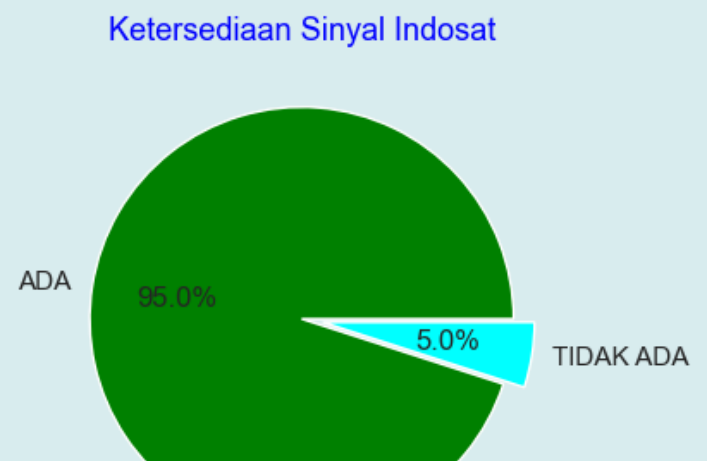

```
labels=df['Ketersediaan_sinyal_telkomsel'].value_counts().index
colors=['green','cyan']
explode=[0,0.1]
values=df['Ketersediaan_sinyal_telkomsel'].value_counts().values

#visualization
plt.figure(figsize=(4,4))
plt.pie(values,explode=explode,labels=labels,colors=colors,autopct='%1.1f%%')
plt.title('Ketersediaan Sinyal Telkomsel',color='Blue',fontsize=13)
plt.show()
```



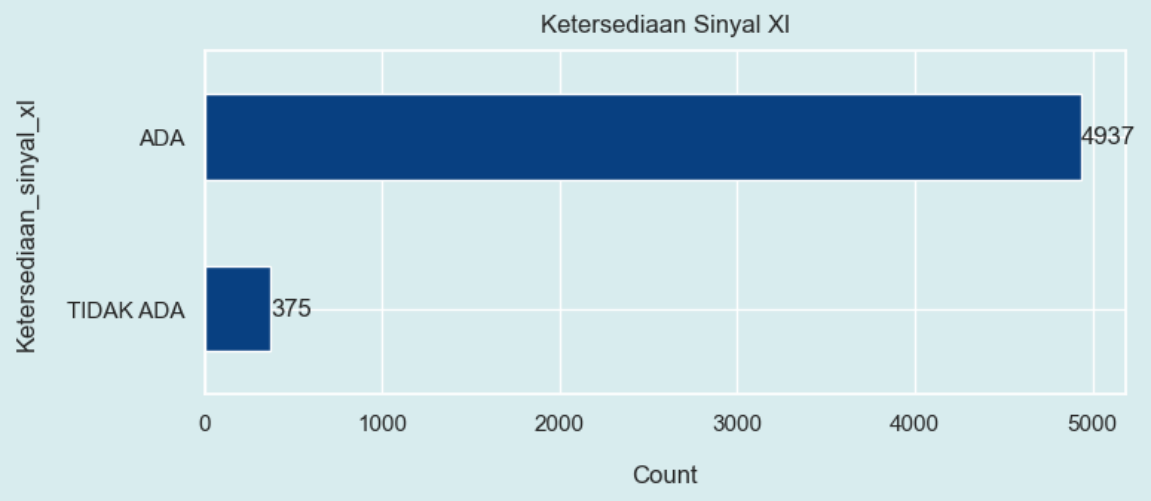

```
dx = df['Ketersediaan_sinyal_indosat'].value_counts().head().sort_values().plot(kind='barh', figsize=(8,3),
dx.bar_label(dx.containers[0], )
plt.xlabel("Count", labelpad=14)
plt.ylabel("Ketersediaan_sinyal_indosat", labelpad=14)
plt.title("Ketersediaan Sinyal Indosat", y=1.02);
```



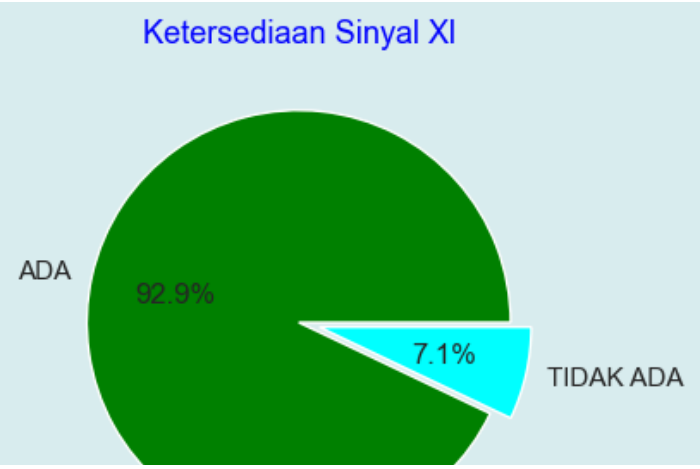

```
labels=df['Ketersediaan_sinyal_indosat'].value_counts().index
colors=['green','cyan']
explode=[0,0.1]
values=df['Ketersediaan_sinyal_indosat'].value_counts().values

#visualization
plt.figure(figsize=(4,4))
plt.pie(values,explode=explode,labels=labels,colors=colors,autopct='%1.1f%%')
plt.title('Ketersediaan Sinyal Indosat',color='Blue',fontsize=13)
plt.show()
```



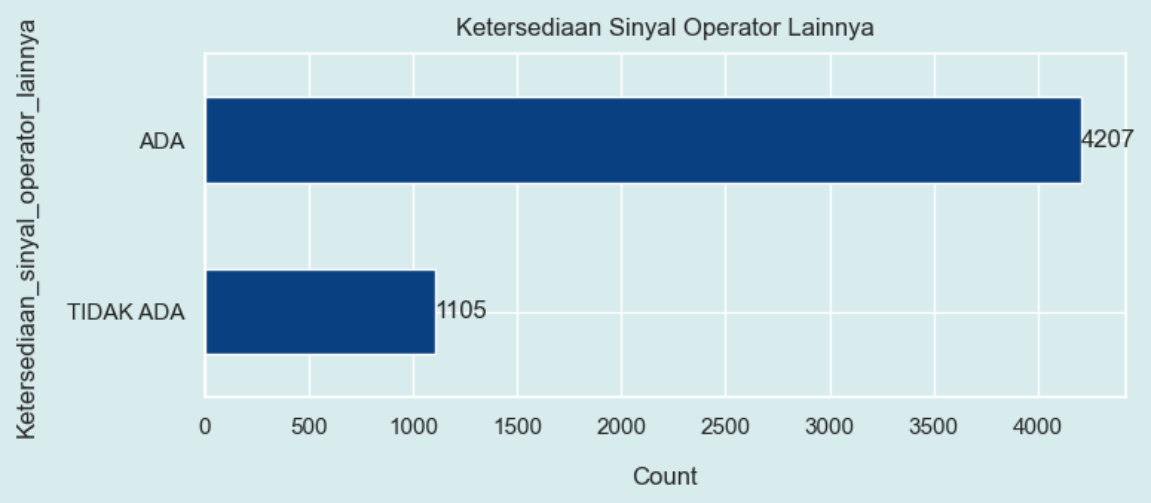

```
dx = df['Ketersediaan_sinyal_xl'].value_counts().head().sort_values().plot(kind='barh', figsize=(8,3), cmap
dx.bar_label(dx.containers[0], )
plt.xlabel("Count", labelpad=14)
plt.ylabel("Ketersediaan_sinyal_xl", labelpad=14)
plt.title("Ketersediaan Sinyal Xl", y=1.02);
```



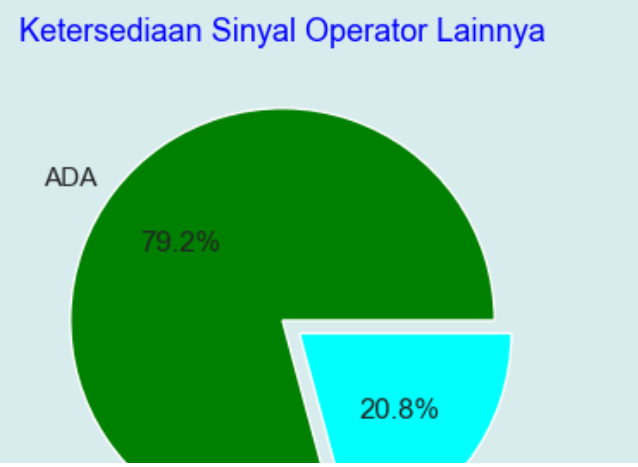

```
labels=df['Ketersediaan_sinyal_xl'].value_counts().index
colors=['green','cyan']
explode=[0,0.1]
values=df['Ketersediaan_sinyal_xl'].value_counts().values

#visualization
plt.figure(figsize=(4,4))
plt.pie(values,explode=explode,labels=labels,colors=colors,autopct='%1.1f%%')
plt.title('Ketersediaan Sinyal Xl',color='Blue',fontsize=13)
plt.show()
```



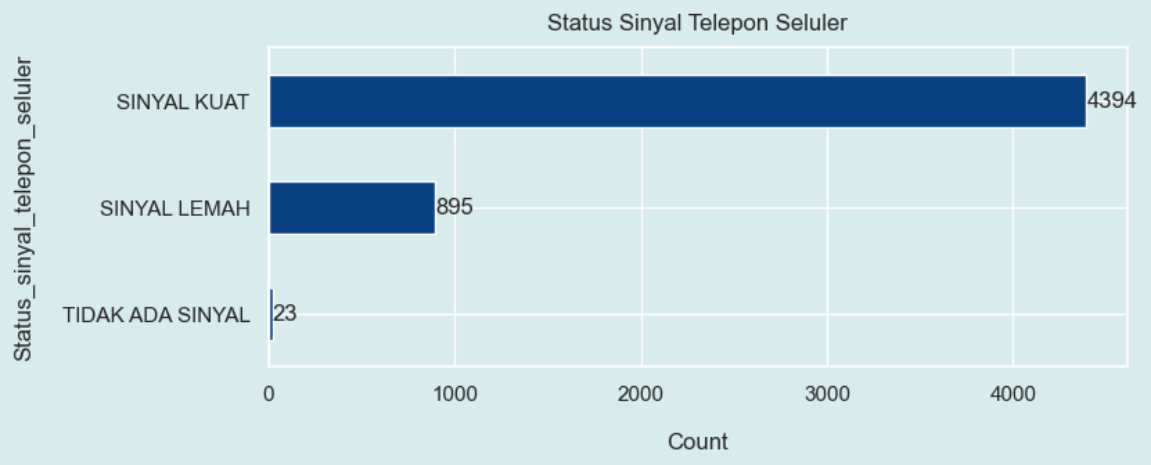

```
dx = df['Ketersediaan_sinyal_operator_lainnya'].value_counts().head().sort_values().plot(kind='barh', figsi
dx.bar_label(dx.containers[0], )
plt.xlabel("Count", labelpad=14)
plt.ylabel("Ketersediaan_sinyal_operator_lainnya", labelpad=14)
plt.title("Ketersediaan Sinyal Operator Lainnya", y=1.02);
```



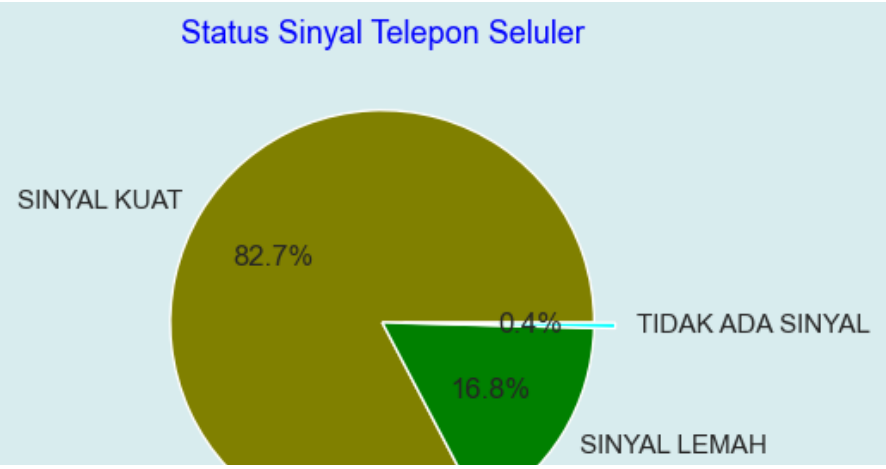

```
labels=df['Ketersediaan_sinyal_operator_lainnya'].value_counts().index
colors=['green','cyan']
explode=[0,0.1]
values=df['Ketersediaan_sinyal_operator_lainnya'].value_counts().values

#visualization
plt.figure(figsize=(4,4))
plt.pie(values,explode=explode,labels=labels,colors=colors,autopct='%1.1f%%')
plt.title('Ketersediaan Sinyal Operator Lainnya',color='Blue',fontsize=13)
plt.show()
```



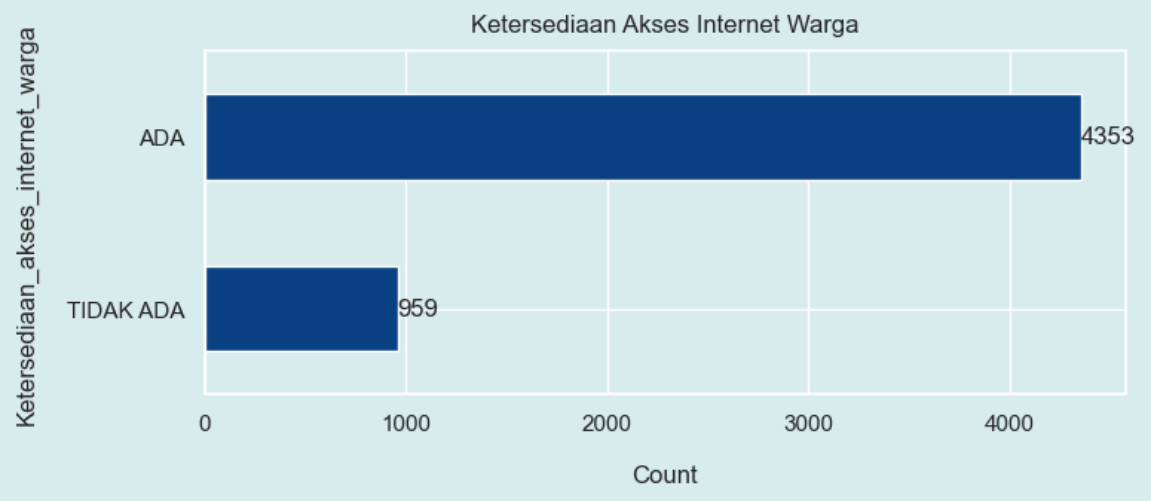

```
dx = df['Status_sinyal_telepon_seluler'].value_counts().head().sort_values().plot(kind='barh', figsize=(8,3
dx.bar_label(dx.containers[0], )
plt.xlabel("Count", labelpad=14)
plt.ylabel("Status_sinyal_telepon_seluler", labelpad=14)
plt.title("Status Sinyal Telepon Seluler", y=1.02);
```



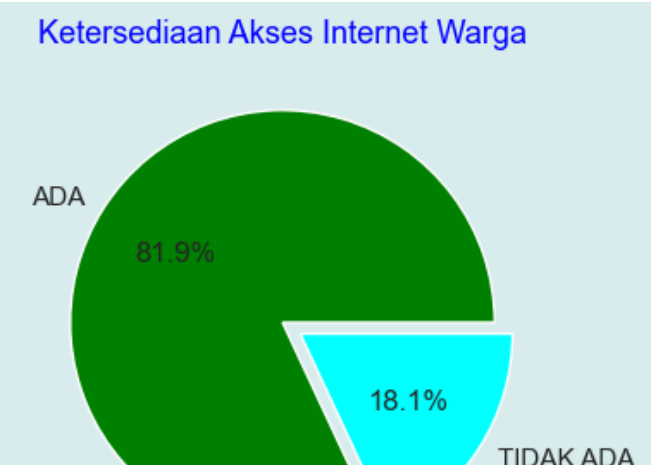

```
labels=df['Status_sinyal_telepon_seluler'].value_counts().index
colors=['olive','green', 'cyan']
explode=[0,0,0.1]
values=df['Status_sinyal_telepon_seluler'].value_counts().values

#visualization
plt.figure(figsize=(4,4))
plt.pie(values,explode=explode,labels=labels,colors=colors,autopct='%1.1f%%')
plt.title('Status Sinyal Telepon Seluler',color='Blue',fontsize=13)
plt.show()
```




```
dx = df['Ketersediaan_akses_internet_warga'].value_counts().head().sort_values().plot(kind='barh', figsize=
dx.bar_label(dx.containers[0], )
plt.xlabel("Count", labelpad=14)
plt.ylabel("Ketersediaan_akses_internet_warga", labelpad=14)
plt.title("Ketersediaan Akses Internet Warga", y=1.02);
```

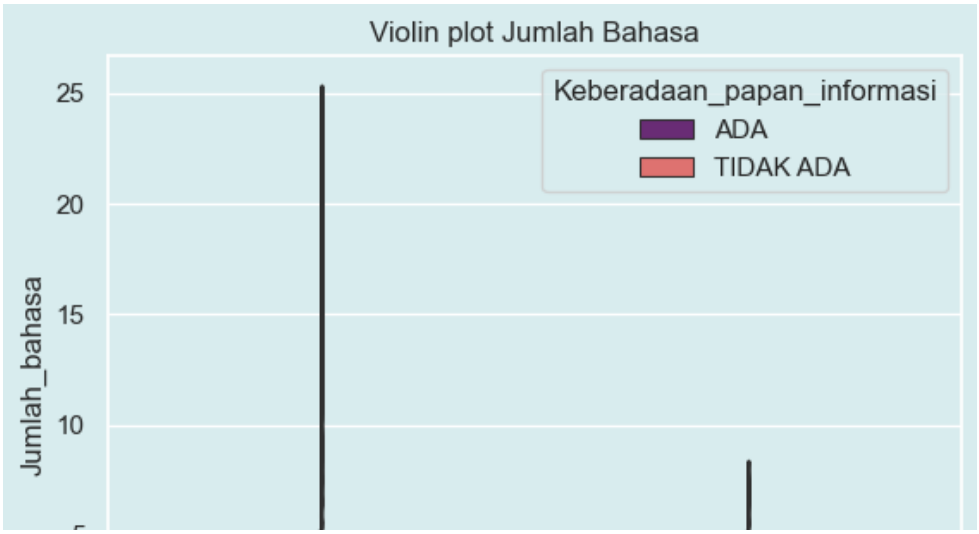



```
labels=df['Ketersediaan_akses_internet_warga'].value_counts().index
colors=['green','cyan']
explode=[0,0.1]
values=df['Ketersediaan_akses_internet_warga'].value_counts().values

#visualization
plt.figure(figsize=(4,4))
plt.pie(values,explode=explode,labels=labels,colors=colors,autopct='%1.1f%%')
plt.title('Ketersediaan Akses Internet Warga',color='Blue',fontsize=13)
plt.show()
```



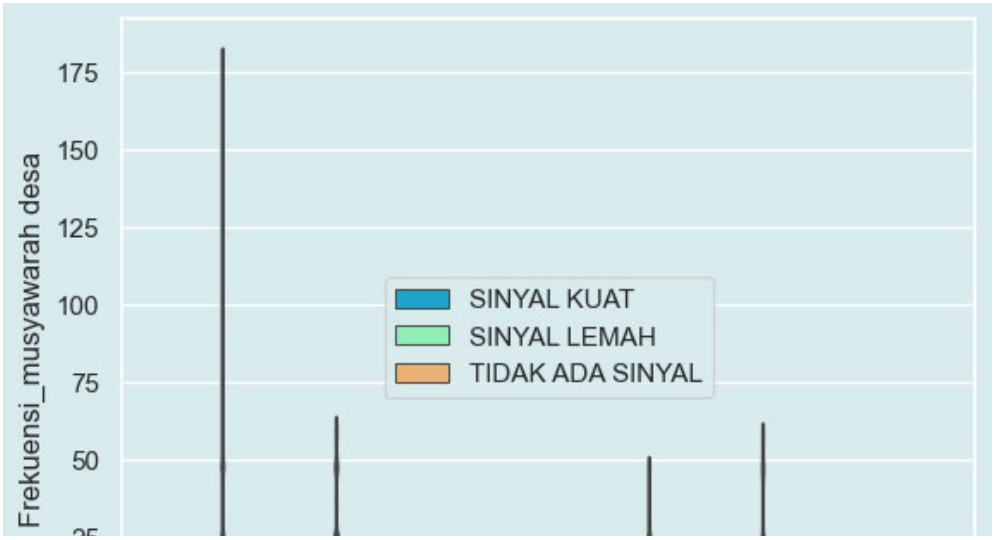

```
sns.violinplot(x=df['Ketersediaan_akses_internet_warga'],y=df['Frekuensi_musyawarah desa'])
plt.show()
```


```
sns.violinplot(x='Ketersediaan_akses_internet_warga', y='Jumlah_bahasa',hue='Keberadaan_papan_informasi',da
plt.title('Violin plot Jumlah Bahasa');
```

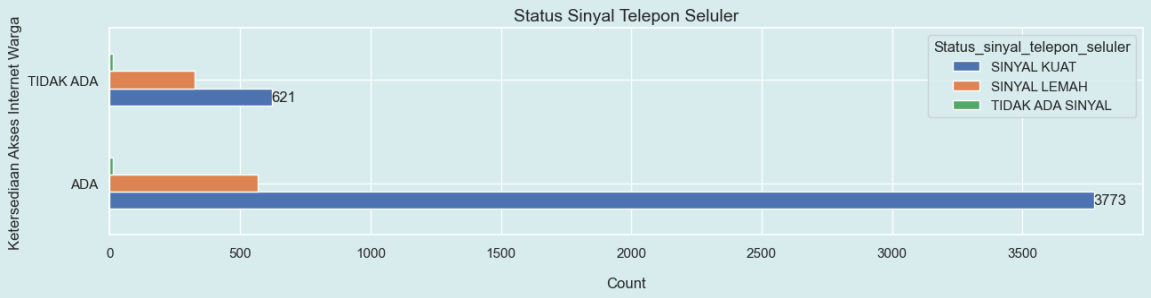



```
sns.violinplot(df['Ketersediaan_akses_internet_warga'],y=df['Frekuensi_musyawarah desa'],hue=df['Status_sin
plt.legend(loc=10)
plt.show()
```

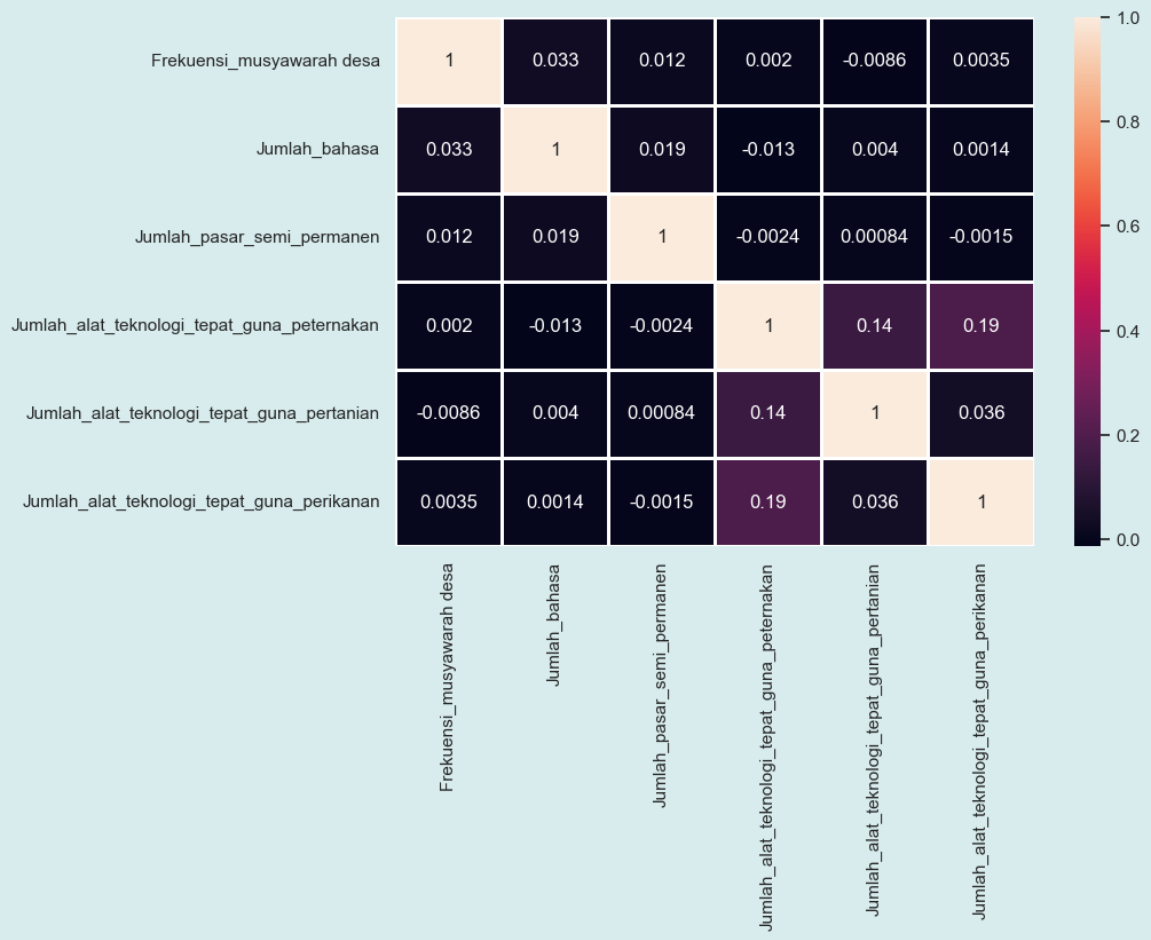
```
C:\Users\Dwita Nurwahyuni\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
  warnings.warn(
```



```
df_bar = (
    df.groupby("Status_sinyal_telepon_seluler")["Ketersediaan_akses_internet_warga"]
    .value_counts()
    .sort_index()
    .unstack()
)
ax = df_bar.T.plot(kind="barh", figsize=(15, 3), )
ax.bar_label(ax.containers[0], )
ax.set_title("Status Sinyal Telepon Seluler", fontsize=14)
ax.set_ylabel("Ketersediaan Akses Internet Warga")
ax.set_xlabel("Count", labelpad=14)
plt.show()
```




```
# heatmap
plt.figure(figsize = (9, 6))
corr = df.corr()
sns.heatmap(data = corr, annot = True, fmt = '.2g', linewidth = 1)
plt.show()
```




```
df.nunique()
```

```
Kabupaten/Kota                              19
Kecamatan                                  501
Kelurahan                                 3400
Keberadaan_papan_informasi                   2
Keberadaan_website_desa                      2
Keberadaan_sarana_informasi_lainnya          2
Frekuensi_musyawarah desa                   50
Jumlah_bahasa                               11
Jumlah_pasar_semi_permanen                  69
Jumlah_alat_teknologi_tepat_guna_peternakan    144
Jumlah_alat_teknologi_tepat_guna_pertanian     126
Jumlah_alat_teknologi_tepat_guna_perikanan     141
Ketersediaan_sinyal_telkomsel                2
Ketersediaan_sinyal_indosat                  2
Ketersediaan_sinyal_xl                       2
Ketersediaan_sinyal_operator_lainnya         2
Status_sinyal_telepon_seluler                3
Ketersediaan_akses_internet_warga            2
dtype: int64
```

```
# dropping columns which are not necessary for prediction

to_drop = ['Kabupaten/Kota','Kecamatan','Kelurahan']

df.drop(to_drop, inplace = True, axis = 1)


df.head()
```

	Keberadaan_papan_informasi	Keberadaan_website_desa	Keberadaan_sarana_informasi_lai
0	ADA	TIDAK ADA	
1	ADA	ADA	
2	ADA	ADA	
3	ADA	ADA	
4	ADA	TIDAK ADA	

```
# checking for multicollinearity

plt.figure(figsize = (9, 6))

corr = df.corr()
mask = np.triu(np.ones_like(corr, dtype = bool))

sns.heatmap(data = corr, mask = mask, annot = True, fmt = '.2g', linewidth = 1)
plt.show()
```

Dari plot diatas dapat dilihat bahwa antar variabel tidak terjadi multikolinieritas dan korelasi tertinggi bernilai dibawah 0.2

```
# separating the feature and target columns

X = df.drop('Ketersediaan_akses_internet_warga', axis = 1)
y = df['Ketersediaan_akses_internet_warga']
```

Encoding Categorical columns

```
# extracting categorical columns
cat_df = X.select_dtypes(include = ['object'])


# printing unique values of each column
for col in cat_df.columns:
    print(f"{col}: \n{cat_df[col].unique()}\n")
```

```
 Keberadaan_papan_informasi:
 ['ADA' 'TIDAK ADA']

 Keberadaan_website_desa:
```

```
['TIDAK ADA' 'ADA']

Keberadaan_sarana_informasi_lainnya:
['ADA' 'TIDAK ADA']

Ketersediaan_sinyal_telkomsel:
['ADA' 'TIDAK ADA']

Ketersediaan_sinyal_indosat:
['ADA' 'TIDAK ADA']

Ketersediaan_sinyal_xl:
['ADA' 'TIDAK ADA']

Ketersediaan_sinyal_operator_lainnya:
['ADA' 'TIDAK ADA']

Status_sinyal_telepon_seluler:
['SINYAL KUAT' 'SINYAL LEMAH' 'TIDAK ADA SINYAL']
```

```
cat_df = pd.get_dummies(cat_df, drop_first = True)
```

```
cat_df.head()
```

	Keberadaan_papan_informasi_TIDAK ADA	Keberadaan_website_desa_TIDAK ADA	Keberadaan_sarana_i
0	0	1	
1	0	0	
2	0	0	
3	0	0	
4	0	1	

```
# extracting the numerical columns

num_df = X.select_dtypes(include = ['int64'])
```

```
num_df.head()
```

	Frekuensi_musyawarah desa	Jumlah_bahasa	Jumlah_pasar_semi_permanen	Jumlah_alat_teknolo
0	36	5	0	
1	6	3	0	
2	15	8	1	
3	4	4	2	
4	6	3	0	

```
num_df.head()
```

	Frekuensi_musyawarah desa	Jumlah_bahasa	Jumlah_pasar_semi_permanen	Jumlah_alat_teknolo
0	36	5	0	
1	6	3	0	
2	15	8	1	
3	4	4	2	
4	6	3	0	

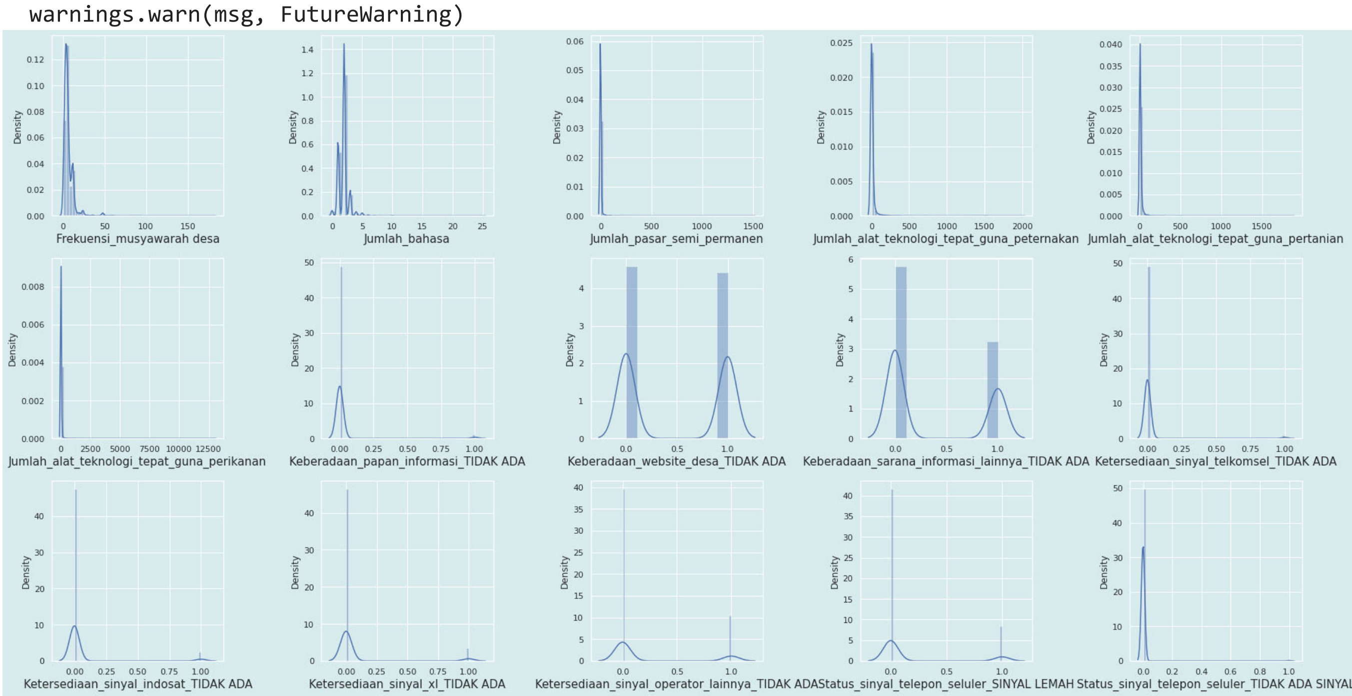
```
# combining the Numerical and Categorical dataframes to get the final dataset

X = pd.concat([num_df, cat_df], axis = 1)
```

```
X.head()
```

	Frekuensi_musyawarah desa	Jumlah_bahasa	Jumlah_pasar_semi_permanen	Jumlah_alat_teknolo
0	36	5	0	
1	6	3	0	
2	15	8	1	
3	4	4	2	
4	6	3	0	

```
plt.figure(figsize = (25, 20))
plotnumber = 1

for col in X.columns:
    if plotnumber <= 15:
        ax = plt.subplot(5, 5, plotnumber)
        sns.distplot(X[col])
        plt.xlabel(col, fontsize = 15)

    plotnumber += 1

plt.tight_layout()
plt.show()
```

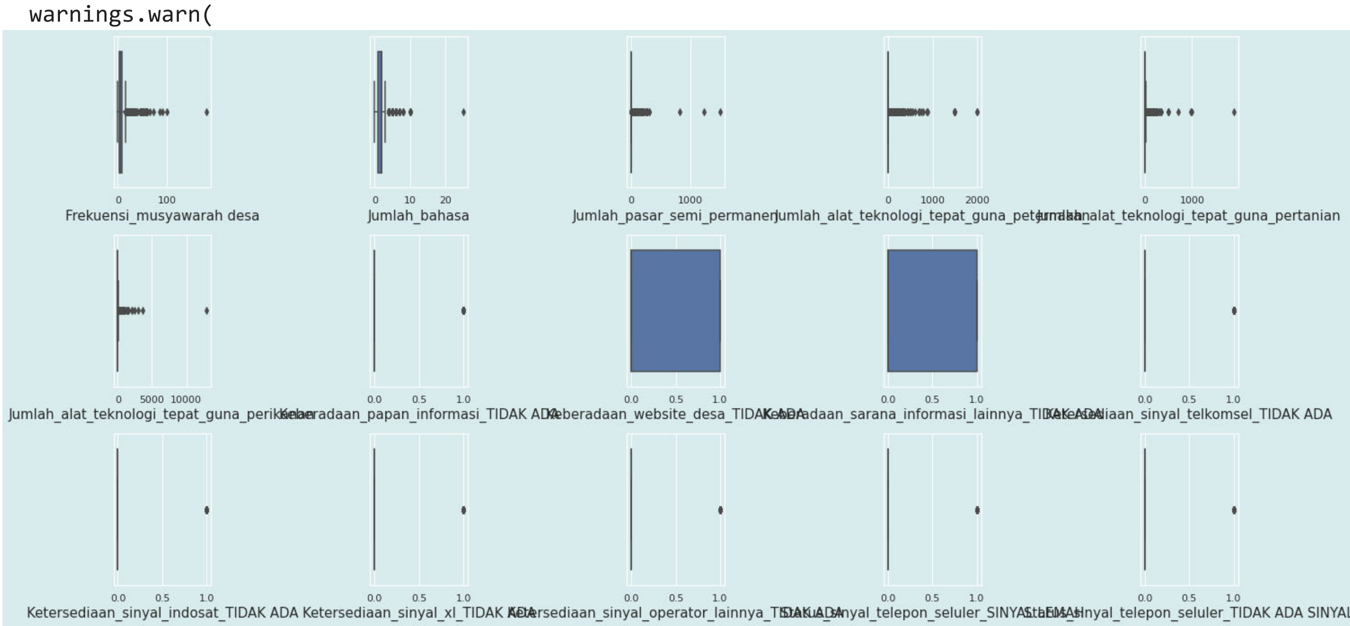
```
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a de
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a de
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a de
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a de
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a de
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a de
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a de
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a de
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a de
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a de
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a de
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a de
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a de
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a de
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a de
  warnings.warn(msg, FutureWarning)
```



Outliers Detection

```
plt.figure(figsize = (20, 15))
plotnumber = 1

for col in X.columns:
    if plotnumber <= 15:
        ax = plt.subplot(5, 5, plotnumber)
        sns.boxplot(X[col])
        plt.xlabel(col, fontsize = 15)

    plotnumber += 1
plt.tight_layout()
plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following var
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following var
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following var
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following var
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following var
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following var
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following var
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following var
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following var
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following var
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following var
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following var
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following var
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following var
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following var
  warnings.warn(
```



Outlier terdapat di beberapa variabel numerik, Tahap selanjutnya akan Dilakukan penyamaan sekala variabel numerik tersebut

Scaling the Data

data Z diskala ulang sehingga μ = 0 dan **σ** = 1, dan dilakukan melalui rumus ini:

$$z = \frac{x_i - \mu}{\sigma}$$

```
# splitting data into training set and test set

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25)
```

```
X_train.head()
```

	Frekuensi_musyawarah desa	Jumlah_bahasa	Jumlah_pasar_semi_permanen	Jumlah_alat_tekn
1234	2	1	0	
1574	12	2	0	
1489	12	1	0	
5180	5	2	0	
244	10	2	0	

```
num_df = X_train[['Frekuensi_musyawarah desa', 'Jumlah_bahasa', 'Jumlah_pasar_semi_permanen',
       'Jumlah_alat_teknologi_tepat_guna_peternakan', 'Jumlah_alat_teknologi_tepat_guna_pertanian', 'Jumlah
       ]]
```

```
# Scaling the numeric values in the dataset

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaled_data = scaler.fit_transform(num_df)
```

```
scaled_num_df = pd.DataFrame(data = scaled_data, columns = num_df.columns, index = X_train.index)
scaled_num_df.head()
```

	Frekuensi_musyawarah desa	Jumlah_bahasa	Jumlah_pasar_semi_permanen	Jumlah_alat_tekn
1234	-0.665861	-0.952266	-0.085714	
1574	0.728363	0.123692	-0.085714	
1489	0.728363	-0.952266	-0.085714	
5180	-0.247594	0.123692	-0.085714	
244	0.449518	0.123692	-0.085714	

```
X_train.drop(columns = scaled_num_df.columns, inplace = True)

X_train = pd.concat([scaled_num_df, X_train], axis = 1)

X_train.head()
```

	Frekuensi_musyawarah desa	Jumlah_bahasa	Jumlah_pasar_semi_permanen	Jumlah_alat_tekn
1234	-0.665861	-0.952266	-0.085714	
1574	0.728363	0.123692	-0.085714	
1489	0.728363	-0.952266	-0.085714	
5180	-0.247594	0.123692	-0.085714	
244	0.449518	0.123692	-0.085714	

Models

Suport Vector Classifier

SVM adalah algoritma supervised yang digunakan untuk klasifikasi, regresi, dan anomali atau pendeteksian pencilan (outlier). SVM berusaha menemukan garis pemisah (hyperplane) yang terbaik pada ruang input space. Prinsip dasar SVM adalah linear classifier, mengingat masalah klasifikasi biner, jika kita memiliki data latihan yang tiap titik data atau observasi tergolong pada kelas yang spesifik, algoritma SVM bisa dilatih berdasarkan data tersebut dan bisa menetapkan titik data yang akan datang ke dalam salah satu dari dua kelas kategori. Namun, SVM juga bisa melakukan klasifikasi non-linier dengan pendekatan menarik yang dikenal dengan sebutan trik kernel, dimana fungsi kernel digunakan untuk mengoperasikan ruang fitur berdimensi tinggi yang terpisah secara non-linier. Algoritma SVM mengambil satu set titik data latihan dan mendirikan sebuah hyperplane dari koleksi hyperplane untuk ruang fitur berdimensi tinggi. Semakin besar batas hyperplane, semakin bagus pemisahannya, dan dengan demikian hal ini akan mengurangi kesalahan generalisasi alat yang mengklasifikasi (classifier) (Sarkar, 2016). Garis pemisah (hyperplane) terbaik antara kedua kelas didapatkan dengan cara mengukur margin hyperplane tersebut dan mencari titik maksimalnya. Margin adalah jarak antara hyperplane tersebut dengan titik terdekat dari masing-masing kelas. Titik yang terpotong oleh supporting hyperplane disebut support vector

```
from sklearn.svm import SVC

svc = SVC()
svc.fit(X_train, y_train)

y_pred = svc.predict(X_test)
```

```
# accuracy_score, confusion_matrix and classification_report

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

svc_train_acc = accuracy_score(y_train, svc.predict(X_train))
svc_test_acc = accuracy_score(y_test, y_pred)

print(f"Training accuracy of Support Vector Classifier is : {svc_train_acc}")
print(f"Test accuracy of Support Vector Classifier is : {svc_test_acc}")

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Training accuracy of Support Vector Classifier is : 0.8323293172690763
Test accuracy of Support Vector Classifier is : 0.8162650602409639
[[1084    0]
 [ 244    0]]
              precision    recall  f1-score   support

         ADA       0.82      1.00      0.90      1084
   TIDAK ADA       0.00      0.00      0.00       244

    accuracy                           0.82      1328
   macro avg       0.41      0.50      0.45      1328
weighted avg       0.67      0.82      0.73      1328

/usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning:
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning:
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning:
  _warn_prf(average, modifier, msg_start, len(result))
```

KNN

Nearest Neighbor merupakan metode untuk mengklasifikasikan suatu data baru berdasarkan similaritas atau kemiripan dengan labeled data. Similaritas menggunakan metrik jarak dengan satuan Euclidian. K- Nearest Neigbor adalah metode pengembangan dari NN (Nearest Neighbor). Dimana, algoritma ini termasuk ke dalam algoritma supervised learning dimana hasil dari instance yang baru diklasifikasikan berdasarkan mayoritas dari kategori K-tetangga terdekat. Tujuan algoritma ini adalah untuk mengklasifikasikan obyek baru berdasarkan atribut dan sampel-sampel dari data training. Algoritma K- Nearest Neigbor menggunakan neighborhood classification sebagai nilai prediksi dari nilai instance yang baru. K- Nearest Neigbor bekerja berdasarkan jarak minimum dari data baru ke data training sampel untuk menentukan K tertangga terdekat. Selanjutnya kita dapatkan nilai mayoritas sebagai hasil prediksi dari data yang baru tersebut. Berikut Gambar 2.1 langkah-langkah dalam metode K-NN.


```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors = 30)
knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)


# accuracy_score, confusion_matrix and classification_report

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

knn_train_acc = accuracy_score(y_train, knn.predict(X_train))
knn_test_acc = accuracy_score(y_test, y_pred)

print(f"Training accuracy of KNN is : {knn_train_acc}")
print(f"Test accuracy of KNN is : {knn_test_acc}")

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
     Training accuracy of KNN is : 0.8335843373493976
     Test accuracy of KNN is : 0.7688253012048193
     [[996  66]
      [241  25]]
                   precision    recall  f1-score   support

              ADA       0.81      0.94      0.87      1062
        TIDAK ADA       0.27      0.09      0.14       266

         accuracy                           0.77      1328
        macro avg       0.54      0.52      0.50      1328
     weighted avg       0.70      0.77      0.72      1328
```

Decision Tree Classifier

Decision Tree merupakan salah satu metode klasifikasi yang menggunakan representasi struktur pohon (tree). Setiap node merepresentasikan atribut, cabang (branche) dari node merepresentasikan nilai dari atribut, daun (leave) merepresentasikan kelas. Node yang paling atas disebut dengan akar atau root. Algoritma decision tree sendiri didasarkan pada pendekatan divide-and-conquer sebagai klasifikasi suatu permasalahan. Algoritma tersebut berbentuk top-down, dengan mencari setiap tahap atribut untuk diklasifikasikan pada bagian class terbaik, dan memproses submasalah secara rekursif dari hasil klasifikasi tersebut. Strategi inilah yang menghasilkan decision tree dimana hal ini dapat diubah menjadi satu set classification rules.

https://scikit-learn.org/stable/modules/tree.html#decision-trees

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

```
from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)

y_pred = dtc.predict(X_test)


# accuracy_score, confusion_matrix and classification_report

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

dtc_train_acc = accuracy_score(y_train, dtc.predict(X_train))
dtc_test_acc = accuracy_score(y_test, y_pred)

print(f"Training accuracy of Decision Tree is : {dtc_train_acc}")
print(f"Test accuracy of Decision Tree is : {dtc_test_acc}")

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
     Training accuracy of Decision Tree is : 0.9929718875502008
     Test accuracy of Decision Tree is : 0.5993975903614458
     [[686 376]
      [156 110]]
                   precision    recall  f1-score   support

              ADA       0.81      0.65      0.72      1062
        TIDAK ADA       0.23      0.41      0.29       266

         accuracy                           0.60      1328
        macro avg       0.52      0.53      0.51      1328
     weighted avg       0.70      0.60      0.63      1328
```

```
# hyper parameter tuning
# criterion : The function to measure the quality of a split.
# max_depth : The maximum depth of the tree.
# min_samples_split : The minimum number of samples required to split an internal node
# min_samples_leaf : The minimum number of samples required to be at a leaf node
# https://scikit-learn.org/stable/modules/tree.html#tree-mathematical-formulation

from sklearn.model_selection import GridSearchCV

grid_params = {
    'criterion' : ['gini', 'entropy'],
    'max_depth' : [3, 5, 7, 10],
    'min_samples_split' : range(2, 10, 1),
    'min_samples_leaf' : range(2, 10, 1)
}

grid_search = GridSearchCV(dtc, grid_params, cv = 5, n_jobs = -1, verbose = 1)
grid_search.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 512 candidates, totalling 2560 fits
GridSearchCV(cv=5, estimator=DecisionTreeClassifier(), n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [3, 5, 7, 10],
                         'min_samples_leaf': range(2, 10),
                         'min_samples_split': range(2, 10)},
             verbose=1)
```

```
# best parameters and best score

print(grid_search.best_params_)
print(grid_search.best_score_)
```

```
{'criterion': 'entropy', 'max_depth': 7, 'min_samples_leaf': 2, 'min_samples_split': 2}
0.8258062583935992
```

```
# best estimator

dtc = grid_search.best_estimator_

y_pred = dtc.predict(X_test)


# accuracy_score, confusion_matrix and classification_report

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

dtc_train_acc = accuracy_score(y_train, dtc.predict(X_train))
dtc_test_acc = accuracy_score(y_test, y_pred)

print(f"Training accuracy of Decision Tree is : {dtc_train_acc}")
print(f"Test accuracy of Decision Tree is : {dtc_test_acc}")

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Training accuracy of Decision Tree is : 0.848644578313253
Test accuracy of Decision Tree is : 0.6197289156626506
[[770 292]
 [213  53]]
              precision    recall  f1-score   support
```

```
              ADA       0.78      0.73      0.75      1062
        TIDAK ADA       0.15      0.20      0.17       266

         accuracy                           0.62      1328
        macro avg       0.47      0.46      0.46      1328
     weighted avg       0.66      0.62      0.64      1328
```

Random Forest Classifier

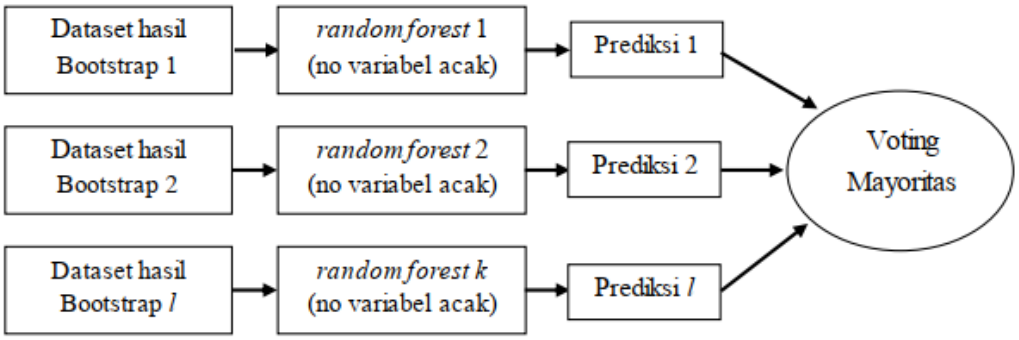
Random Forest merupakan supervised learning algorithm, 'forest' yang dibangun adalah ensemble decision tree. Random Forest membangun banyak decision tree dan menggabungkannya untuk mendapatkan prediksi yang lebih akurat dan stabil. Random Forest bekerja dengan cara membangun lebih dari satu Decision Tree secara random saat training. Hasil yang diberikan oleh Random Forest untuk klasifikasi adalah modus dari decision tree nya.

https://scikit-learn.org/stable/modules/ensemble.html#random-forests

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Metode random forest adalah pengembangn dari metode CART, yaitu dengan menerapkan metode bootstrap aggregating (bagging) dan random feature selection. Dalam random forest, banyak pohon ditumbuhkan sehingga terbentuk hutan (forest), kemudian analisis dilakukan pada kumpulan pohon tersebut. Secara sederhana, algoritma pembentukan random forest dapat disebutkan sebagai berikut. Andaikan data training yang kita miliki berukuran n dan terdiri atas p variabel penjelas (prediktor). Tahapan penyusunan dan pendugaan menggunakan random forest adalah:

1. (Tahapan bootstrap) tarik sampel acak dengan pengembalian berukuran n dari data training.
2. Dengan menggunakan contoh bootstrap, pohon dibangun sampai mencapai ukuran maksimum (tanpa pengembalian). Susun pohon berdasarkan data n tersebut, namun pada setiap proses pemisahan pilih secara acak m < p peubah penjelas, dan dilakukan pemisahan terbaik (tahapan random sub-setting).
3. Ulangi langkah 1-2 sebanyak l kali sehingga terbentuk sebuah hutan yang terdiri atas l pohon
4. Lakukan pendugaan gabungan berdasarkan l buah pohon tersebut (missal menggunakan majority vote untuk kasus klasifikasi atau rata-rata untuk kasus regresi).



Klik dua kali (atau tekan Enter) untuk mengedit

```
from sklearn.ensemble import RandomForestClassifier

rand_clf = RandomForestClassifier(criterion= 'entropy', max_depth= 10, max_features= 'sqrt', min_samples_le
rand_clf.fit(X_train, y_train)

y_pred = rand_clf.predict(X_test)


# accuracy_score, confusion_matrix and classification_report

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

rand_clf_train_acc = accuracy_score(y_train, rand_clf.predict(X_train))
rand_clf_test_acc = accuracy_score(y_test, y_pred)

print(f"Training accuracy of Random Forest is : {rand_clf_train_acc}")
print(f"Test accuracy of Random Forest is : {rand_clf_test_acc}")

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
     Training accuracy of Random Forest is : 0.8692269076305221
     Test accuracy of Random Forest is : 0.7996987951807228
     [[1035   27]
      [ 239   27]]
                   precision    recall  f1-score   support

              ADA       0.81      0.97      0.89      1062
        TIDAK ADA       0.50      0.10      0.17       266

         accuracy                           0.80      1328
        macro avg       0.66      0.54      0.53      1328
     weighted avg       0.75      0.80      0.74      1328
```

Ada Boost Classifier

Sebuah ensemble dari classifier adalah kumpulan dari beberapa classifier yang masing-masing keputusan digabungkan dalam beberapa cara untuk mengklasifikasikan contoh uji. Salah satu metode ensemble adalah boosting yaitu sebuah keluarga ensemble yang meliputi banyak algoritma yang dapat mengubah weak learner menjadi Strong learner dimana Adaptive Boosting (Adaboost) merupakan salah satu algoritma yang populer (Zhou, 2012). Secara umum boosting berfokus pada pembuatan model klasifikasi pada setiap iterasinya. Data yang digunakan dalam menyusun model klasifikasi bergantung pada model klasifikasi sebelumnya dan fokus pada data yang salah prediksi (memberi bobot yang lebih besar untuk data yang salah prediksi atau klasifikasi). Data yang salah prediksi akan diperbaiki terus menerus oleh model klasifikasi selanjutnya. Tujuan akhir dari algoritma boosting adalah menyatukan semua model klasifikasi sehingga diperoleh dari weak learner menjadi model klasifikasi strong yang dapat mengklasifikasikan data secara benar. Adaptive Boosting atau biasa disebut AdaBoost yang pertama kali diperkenalkan Schapire (1999). Pada prisipnya AdaBoost memberikan bobot yang sama kepada tiap pengamatan di awal iterasi, kemudian bobot-bobot tersebut berubah selama proses iterasi. Bobot berubah-ubah sesuai dengan tingat kesalahan klasifikasi dari masing-masing pengamatan. Apabila pengamatan pada data training salah terklasifikasi, maka bobotnya akan lebih besar sehingga peluang pengamatan untuk menjadi data training selanjutnya menjadi besar. Hal ini bertujuan untuk memaksa base classifier untuk lebih fokus pada

pengamatan yang sulit untuk dipelajari. AdaBoost menyusun final classifier dengan mengombinasikan sekumpulan weak classifier diakhir iterasi boosting-nya.

https://scikit-learn.org/stable/modules/ensemble.html#adaboost

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

```
from sklearn.ensemble import AdaBoostClassifier

ada = AdaBoostClassifier(base_estimator = dtc)

parameters = {
    'n_estimators' : [50, 70, 90, 120, 180, 200],
    'learning_rate' : [0.001, 0.01, 0.1, 1, 10],
    'algorithm' : ['SAMME', 'SAMME.R']
}

grid_search = GridSearchCV(ada, parameters, n_jobs = -1, cv = 5, verbose = 1)
grid_search.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 60 candidates, totalling 300 fits
GridSearchCV(cv=5,
             estimator=AdaBoostClassifier(base_estimator=DecisionTreeClassifier(criterion='entropy',
                                                                                max_depth=7,
                                                                                min_samples_leaf=2)),
             n_jobs=-1,
             param_grid={'algorithm': ['SAMME', 'SAMME.R'],
                         'learning_rate': [0.001, 0.01, 0.1, 1, 10],
                         'n_estimators': [50, 70, 90, 120, 180, 200]},
             verbose=1)
```

```
# best parameter and best score

print(grid_search.best_params_)
print(grid_search.best_score_)
```

```
{'algorithm': 'SAMME', 'learning_rate': 0.1, 'n_estimators': 120}
0.8330832329779387
```

```
# best estimator

ada = grid_search.best_estimator_

y_pred = ada.predict(X_test)


# accuracy_score, confusion_matrix and classification_report

ada_train_acc = accuracy_score(y_train, ada.predict(X_train))
ada_test_acc = accuracy_score(y_test, y_pred)

print(f"Training accuracy of Ada Boost is : {ada_train_acc}")
print(f"Test accuracy of Ada Boost is : {ada_test_acc}")

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Training accuracy of Ada Boost is : 0.8551706827309237
Test accuracy of Ada Boost is : 0.7914156626506024
```

```
[[1028   34]
 [ 243   23]]
              precision    recall  f1-score   support

         ADA       0.81      0.97      0.88      1062
   TIDAK ADA       0.40      0.09      0.14       266

    accuracy                           0.79      1328
   macro avg       0.61      0.53      0.51      1328
weighted avg       0.73      0.79      0.73      1328
```

Gradient Boosting Classifier

Boosting merupakan meta-algoritma dalam machine learning untuk melakukan supervised learning. Meta-algoritma adalah algoritma yang menggunakan algoritma lain sebagai perwakilan, dan juga merupakan algoritma yang memiliki sub-algoritma sebagai peubah dan parameter yang dapat diganti. Secara umum, boosting terjadi dalam iterasi, secara incremental menambahkan weak learner ke dalam satu strong learner. Pada setiap iterasi, satu weak learner belajar dari suatu data latihan. Kemudian, weak learner itu ditambahkan ke dalam strong learner. Setelah weak learner ditambahkan, data-data kemudian diubah masing-masing bobotnya. Data-data yang mengalami kesalahan klasifikasi akan mengalami penambahan bobot, dan data-data yang terklasifikasi dengan benar akan mengalami pengurangan bobot. Oleh karena itu, weak learner pada iterasi selanjutnya akan lebih terfokus pada data-data yang mengalami kesalahan klasifikasi oleh weak learner yang sebelumnya. Gradient boosting adalah salah satu dari algoritma boosting, dimana mengasilkan model prediksi dari weak learner berbentuk decision tree. Gradient boosting melatih decision tree untuk meminimalkan loss function, Gradient boosting menangan fitur kategoris, perluasan dari multiclass classification setting, tidak memerlukan fitur scaling, dan dapat menangkap fitur nonlinearities dan interaksi

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html#examples-using-sklearn-ensemble-gradientboostingclassifier

```
from sklearn.ensemble import GradientBoostingClassifier

gb = GradientBoostingClassifier()
gb.fit(X_train, y_train)

# accuracy score, confusion matrix and classification report of gradient boosting classifier

gb_acc = accuracy_score(y_test, gb.predict(X_test))

print(f"Training Accuracy of Gradient Boosting Classifier is {accuracy_score(y_train, gb.predict(X_train))}
print(f"Test Accuracy of Gradient Boosting Classifier is {gb_acc} \n")

print(f"Confusion Matrix :- \n{confusion_matrix(y_test, gb.predict(X_test))}\n")
print(f"Classification Report :- \n {classification_report(y_test, gb.predict(X_test))}")
```

```
Training Accuracy of Gradient Boosting Classifier is 0.8514056224899599
Test Accuracy of Gradient Boosting Classifier is 0.7206325301204819

Confusion Matrix :-
[[892 170]
 [201  65]]

Classification Report :-
```

```
                precision    recall  f1-score   support

           ADA       0.82      0.84      0.83      1062
     TIDAK ADA       0.28      0.24      0.26       266

      accuracy                           0.72      1328
     macro avg       0.55      0.54      0.54      1328
  weighted avg       0.71      0.72      0.71      1328
```

Stochastic Gradient Boosting (SGB)

Stochastic Gradient Boosting (SGB) merupakan modifikasi dari Gradient Boosting (GB) yang termotivasi oleh metode Bootstrap Aggegrating. Friedman sebagai penemu SGB mengatakan bahwa pada setiap iterasi algoritma, base learner harus sesuai pada subsample pelatihan yang diambil secara acak tanpa penggantian dan mengamati peningkatan substansial dalam akurasi GB dengan modifikasi tersebut. SGB dapat mengatasi permasalahan regression sehingga metode ini merupakan jenis supervised learning diperlukan dataset seperti berikut(x, y) i=n1, dimana x = (x1, … , xd) mereferensikan sebagai input variable dan y merupakan label dari variable tersebut. Ketika