# DoE (Design of Experiment)

Disediakan dataset Banknote Authentication yang dapat didownload pada link berikut. Lakukan prediksi apakah suatu data banknote authentic atau forgery (kolom **class**), bernilai 0 jika authentic, dan 1 jika forgery.

Author : Michael Jonathan Halim 13521124

# 0. Loading Data and Library

```
# Put your library here
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split, cross_val_score,
GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, confusion_matrix, classification_report
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler
from sklearn.svm import SVC
from sklearn.ensemble import VotingClassifier
from sklearn.ensemble import StackingClassifier

# Read data here
df = pd.read_csv("data_banknote_authentication.csv")
```

# I. Data Understanding

Tujuan dari bagian ini adalah peserta dapat memahami kualitas dari data yang diberikan. Hal ini meliputi:

1. Ukuran data
2. Statistik dari tiap fitur
3. Pencilan (outlier)
4. Korelasi
5. Distribusi

# I.1

Carilah:

1. Ukuran dari data (instances dan features)
2. Tipe dari tiap-tiap fitur
3. Banyaknya unique values dari fitur yang bertipe kategorikal
4. Nilai minimum, maksimum, rata-rata, median, dan standar deviasi dari fitur yang tidak bertipe kategorikal

```python
# I.1 Put your code here
# 1. Get shape
instances = df.shape[0]
features = df.shape[1]

# Print shape
print(f"Jumlah instances: {instances}")
print(f"Jumlah features: {features}")
```

```
Jumlah instances: 1372
Jumlah features: 5
```

```python
# 2. Tipe dari fitur-fitur
print("Tipe dari tiap-tiap fitur:")
df.dtypes
```

```
Tipe dari tiap-tiap fitur:

variance    float64
skewness    float64
curtosis    float64
entropy     float64
target        int64
dtype: object
```

```python
def unique_values(df):
    categorical_features =
df.select_dtypes(include=['object']).columns
    if len(categorical_features) > 0:
        print("Banyaknya unique values dari fitur yang bertipe
kategorikal:")
        for feature in categorical_features:
            unique_values = df[feature].nunique()
            print(f"{feature}: {unique_values} unique values")
    else:
        print("Tidak ada fitur kategorikal pada dataset")

# 3. Unique Values
unique_values(df)
```

```
Tidak ada fitur kategorikal pada dataset
```

```
# 4. Describe data
df.describe()
```

```
          variance     skewness     curtosis      entropy       target
count  1372.000000  1372.000000  1372.000000  1372.000000  1372.000000
mean      0.433735     1.922353     1.397627    -1.191657     0.444606
std       2.842763     5.869047     4.310030     2.101013     0.497103
min      -7.042100   -13.773100    -5.286100    -8.548200     0.000000
25%      -1.773000    -1.708200    -1.574975    -2.413450     0.000000
50%       0.496180     2.319650     0.616630    -0.586650     0.000000
75%       2.821475     6.814625     3.179250     0.394810     1.000000
max       6.824800    12.951600    17.927400     2.449500     1.000000
```

# I.2

Carilah:

1. Missing values dari tiap fitur
2. Outliers dari tiap fitur (gunakan metode yang kalian ketahui)

```
# I.2 Put your code here
# 1. Missing Values
missing_values = df.isnull().sum()
print("Missing values dari tiap fitur:")
print(missing_values)
```

```
Missing values dari tiap fitur:
variance    0
skewness    0
curtosis    0
entropy     0
target      0
dtype: int64
```

```
# 2. Outliers
def find_outliers_iqr(df):
    # Calculate IQR
    Q1 = df.quantile(0.25)
    Q3 = df.quantile(0.75)
    IQR = Q3 - Q1

    # Calculate lower and upper bound
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Get outliers
    outliers = (df < lower_bound) | (df > upper_bound)
    return outliers
```

```python
def find_outliers(df):
    # Find outliers for every numerical features
    outliers_dict = {}
    numeric_features = df.select_dtypes(include=['int64',
'float64']).columns
    for feature in numeric_features:
        outliers_indices = find_outliers_iqr(df[feature])
        outliers_dict[feature] = df[outliers_indices]

    # Display and remove outliers
    print("Outliers dari tiap fitur (metode IQR - Interquartile
Range):")
    for feature, outliers in outliers_dict.items():
        print(f"{feature}:")
        if len(outliers) > 0:
            print(outliers)
            # Remove outliers from df
            df = df.drop(outliers.index)
        else:
            print("Tidak ada")
        print()

    return df

find_outliers(df)
```

```
Outliers dari tiap fitur (metode IQR - Interquartile Range):
variance:
Tidak ada

skewness:
Tidak ada

curtosis:
     variance  skewness  curtosis   entropy  target
765   -3.8483  -12.8047   15.6824 -1.281000       1
780   -3.5801  -12.9309   13.1779 -2.567700       1
815   -3.1128   -6.8410   10.7402 -1.017200       1
816   -4.8554   -5.9037   10.9818 -0.821990       1
820   -4.0025  -13.4979   17.6772 -3.320200       1
821   -4.0173   -8.3123   12.4547 -1.437500       1
826   -4.2110  -12.4736   14.9704 -1.388400       1
841   -3.8858  -12.8461   12.7957 -3.135300       1
877   -5.1216   -5.3118   10.3846 -1.061200       1
881   -4.4861  -13.2889   17.3087 -3.219400       1
882   -4.3876   -7.7267   11.9655 -1.454300       1
887   -3.2692  -12.7406   15.5573 -0.141820       1
902   -2.8957  -12.0205   11.9149 -2.755200       1
937   -2.9020   -7.6563   11.8318 -0.842680       1
938   -4.3773   -5.5167   10.9390 -0.408200       1
```

```
942    -3.3793  -13.7731   17.9274 -2.032300       1
943    -3.1273   -7.1121   11.3897 -0.083634       1
948    -3.4917  -12.1736   14.3689 -0.616390       1
949    -3.1158   -8.6289   10.4403  0.971530       1
963    -3.3863  -12.9889   13.0545 -2.720200       1
998    -3.0866   -6.6362   10.5405 -0.891820       1
999    -4.7331   -6.1789   11.3880 -1.074100       1
1003   -3.8203  -13.0551   16.9583 -2.305200       1
1004   -3.7181   -8.5089   12.3630 -0.955180       1
1009   -3.5713  -12.4922   14.8881 -0.470270       1
1024   -3.0061  -12.2377   11.9552 -2.160300       1
1059   -3.2305   -7.2135   11.6433 -0.946130       1
1060   -4.8426   -4.9932   10.4052 -0.531040       1
1064   -3.6961  -13.6779   17.5795 -2.618100       1
1065   -3.6012   -6.5389   10.5234 -0.489670       1
1070   -3.1423  -13.0365   15.6773 -0.661650       1
1085   -2.6649  -12.8130   12.6689 -1.908200       1
1120   -3.1875   -7.5756   11.8678 -0.578890       1
1121   -4.6765   -5.6636   10.9690 -0.334490       1
1125   -3.5985  -13.6593   17.6052 -2.492700       1
1126   -3.3582   -7.2404   11.4419 -0.571130       1
1131   -4.0214  -12.8006   15.6199 -0.956470       1
1132   -3.3884   -8.2150   10.3315  0.981870       1
1146   -3.7300  -12.9723   12.9817 -2.684000       1
1181   -3.5895   -6.5720   10.5251 -0.163810       1
1182   -5.0477   -5.8023   11.2440 -0.390100       1
1186   -4.2440  -13.0634   17.1116 -2.801700       1
1187   -4.0218   -8.3040   12.5550 -1.509900       1
1192   -4.4018  -12.9371   15.6559 -1.680600       1
1207   -3.7930  -12.7095   12.7957 -2.825000       1
1243   -5.0676   -5.1877   10.4266 -0.867250       1
1247   -4.4775  -13.0303   17.0834 -3.034500       1
1248   -4.1958   -8.1819   12.1291 -1.601700       1
1253   -4.5531  -12.5854   15.4417 -1.498300       1
1268   -3.9411  -12.8792   13.0597 -3.312500       1
1304   -5.2943   -5.1463   10.3332 -1.118100       1
1308   -4.6338  -12.7509   16.7166 -3.216800       1
1309   -4.2887   -7.8633   11.8387 -1.897800       1
1314   -3.5060  -12.5667   15.1606 -0.752160       1
1329   -2.9672  -13.2869   13.4727 -2.627100       1
1364   -2.8391   -6.6300   10.4849 -0.421130       1
1365   -4.5046   -5.8126   10.8867 -0.528460       1
1369   -3.7503  -13.4586   17.5932 -2.777100       1
1370   -3.5637   -8.3827   12.3930 -1.282300       1

entropy:
      variance  skewness  curtosis  entropy  target
41    -0.20620    9.2207 -3.704400  -6.8103       0
45    -0.78690    9.5663 -3.786700  -7.5034       0
```

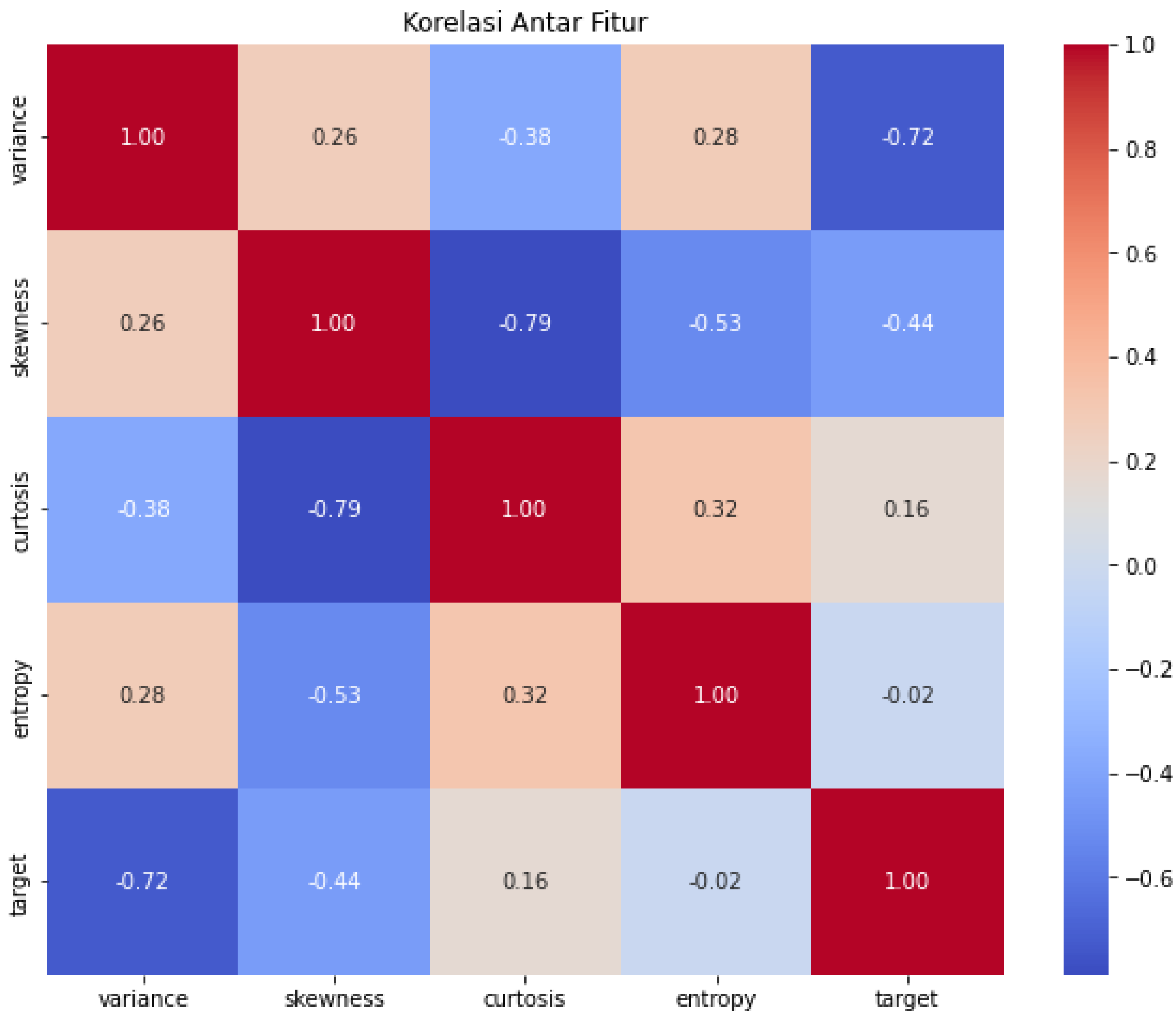
```
47    -0.78690    9.5663 -3.786700  -7.5034       0
59    -0.78289   11.3603 -0.376440  -7.0495       0
139   -0.20620    9.2207 -3.704400  -6.8103       0
194   -2.34100   12.3784  0.704030  -7.5836       0
202   -0.78689    9.5663 -3.786700  -7.5034       0
291   -2.21530   11.9625  0.078538  -7.7853       0
341   -1.18040   11.5093  0.155650  -6.8194       0
394   -2.26230   12.1177  0.288460  -7.7581       0
465   -2.69890   12.1984  0.676610  -8.5482       0
529   -1.38850   12.5026  0.691180  -7.5487       0
543   -1.42170   11.6542 -0.057699  -7.1025       0
562   -2.46040   12.7302  0.917380  -7.6418       0
581   -1.96670   11.8052 -0.404720  -7.8719       0
606   -1.42750   11.8797  0.416130  -6.9978       0
615   -0.20620    9.2207 -3.704400  -6.8103       0
740   -2.44730   12.6247  0.735730  -7.6612       0
776   -5.90340    6.5679  0.676610  -6.6797       1
791   -4.47790    7.3708 -0.312180  -6.7754       1
837   -6.28150    6.6651  0.525810  -7.0107       1
852   -4.88610    7.0542 -0.172520  -6.9590       1
898   -5.24060    6.6258 -0.199080  -6.8607       1
974   -5.03010    7.5032 -0.133960  -7.5034       1
1142  -6.57730    6.8017  0.854830  -7.5344       1
1157  -5.20490    7.2590  0.070827  -7.3004       1
1164  -6.33640    9.2848  0.014275  -6.7844       1
1203  -6.73870    6.9879  0.678330  -7.5887       1
1218  -5.44140    7.2363  0.109380  -7.5642       1
1225  -6.52350    9.6014 -0.253920  -6.9642       1
1264  -6.65100    6.7934  0.686040  -7.5887       1
1279  -5.30120    7.3915  0.029699  -7.3987       1
1286  -6.42470    9.5311  0.022844  -6.8517       1

target:
Tidak ada


      variance  skewness  curtosis  entropy  target
0      3.62160   8.66610  -2.80730 -0.44699       0
1      4.54590   8.16740  -2.45860 -1.46210       0
2      3.86600  -2.63830   1.92420  0.10645       0
3      3.45660   9.52280  -4.01120 -3.59440       0
4      0.32924  -4.45520   4.57180 -0.98880       0
...        ...       ...       ...      ...     ...
1363  -1.16670  -1.42370   2.92410  0.66119       1
1366  -2.41000   3.74330  -0.40215 -1.29530       1
1367   0.40614   1.34920  -1.45010 -0.55949       1
1368  -1.38870  -4.87730   6.47740  0.34179       1
1371  -2.54190  -0.65804   2.68420  1.19520       1

[1280 rows x 5 columns]
```

# I.3

Carilah:

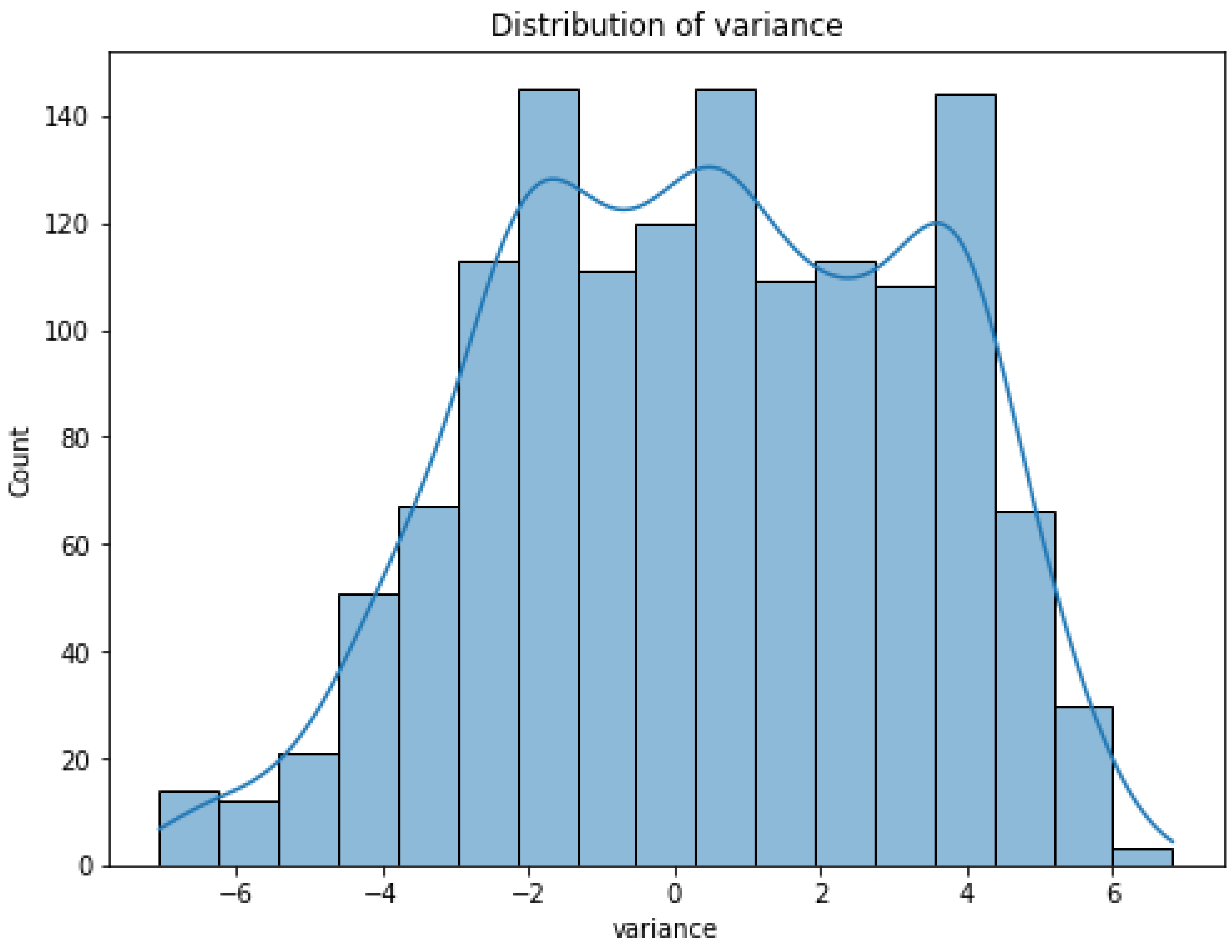
1. Korelasi antar fitur
2. Visualisasikan distribusi dari tiap fitur (kategorikal dan kontinu)
3. Visualisasikan distribusi dari tiap fitur, dengan data dibagi tiap unique values fitur survived

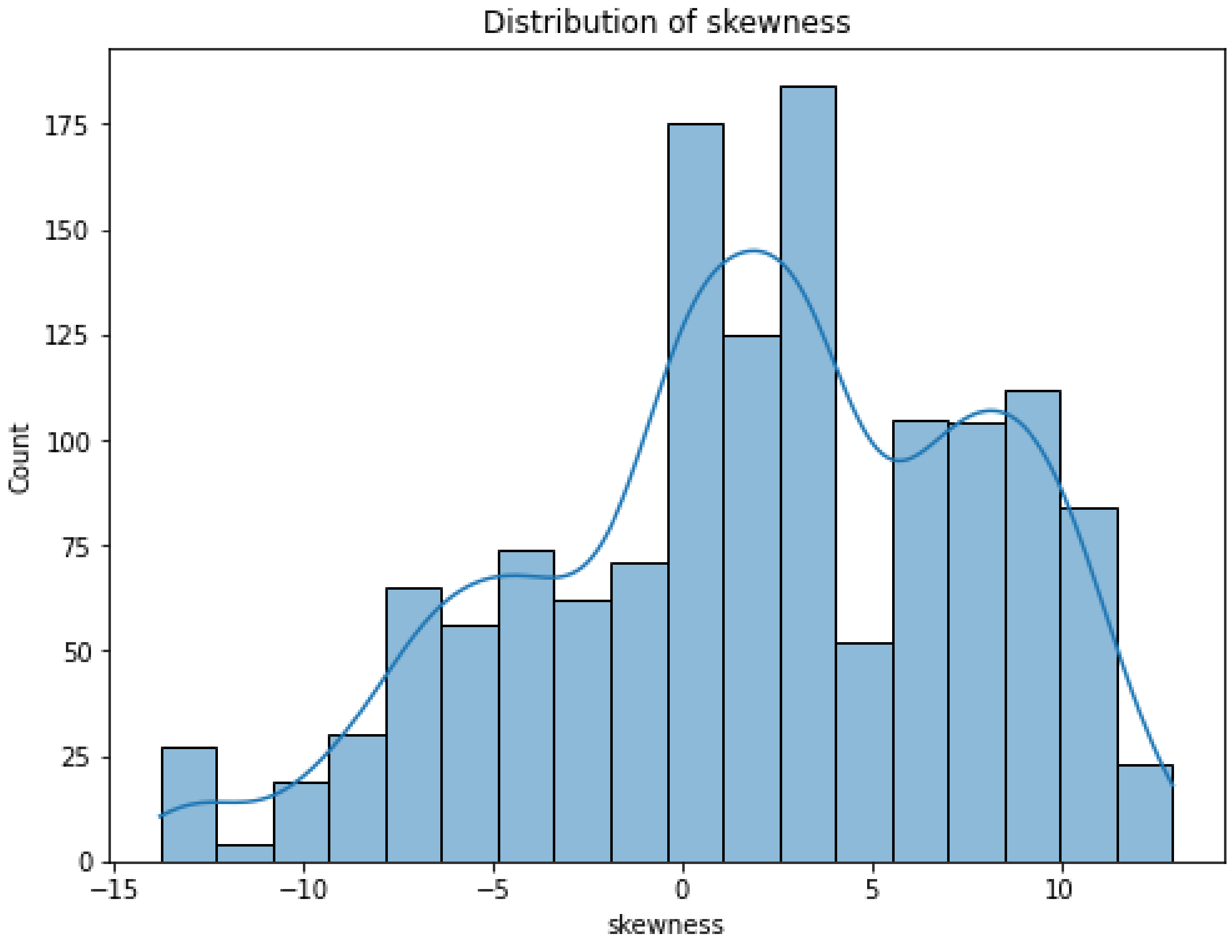
```
# I.3 Put your code here
# 1. Korelasi antar fitur
correlation_matrix = df.corr()

# Plot matriks korelasi
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
fmt=".2f")
plt.title('Korelasi Antar Fitur')
plt.show()
```

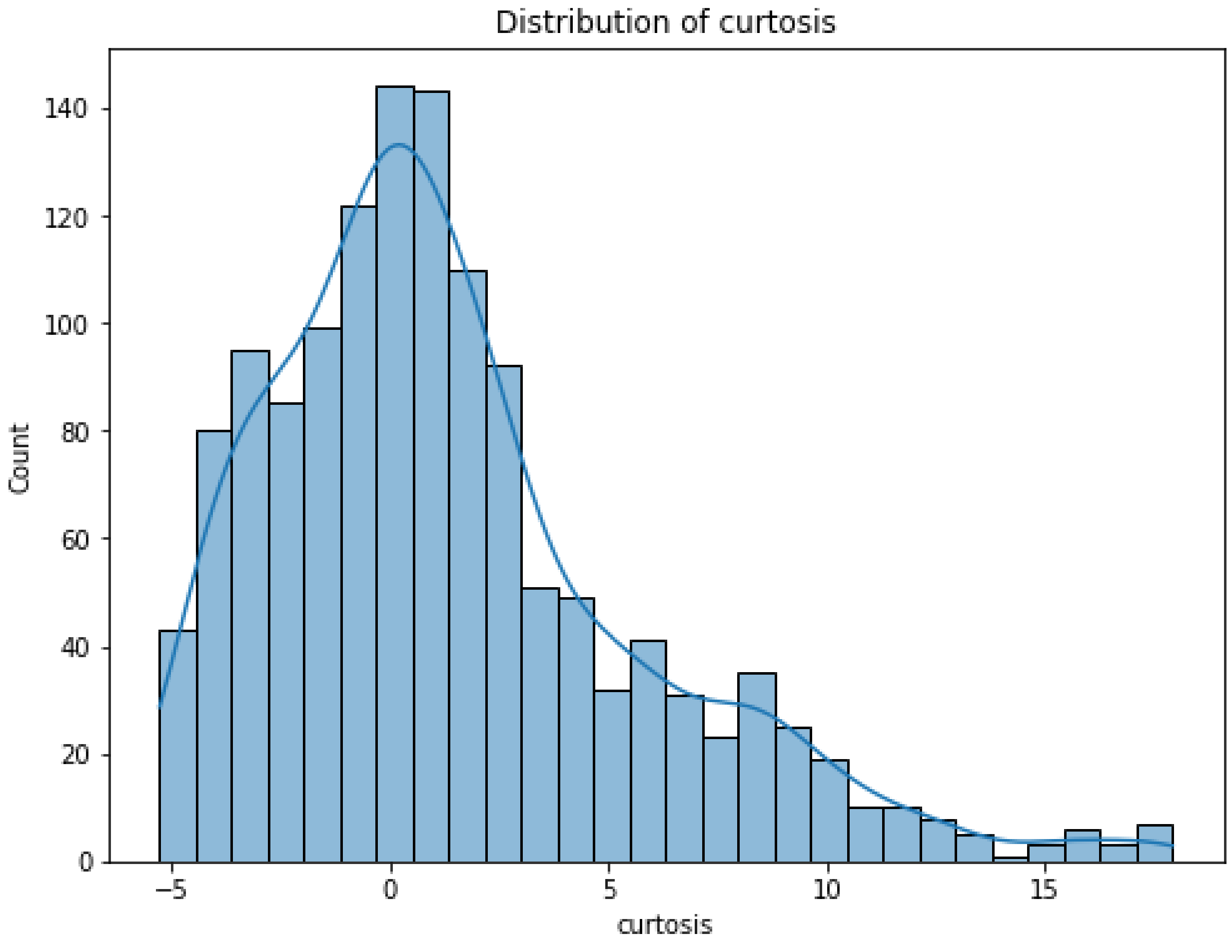


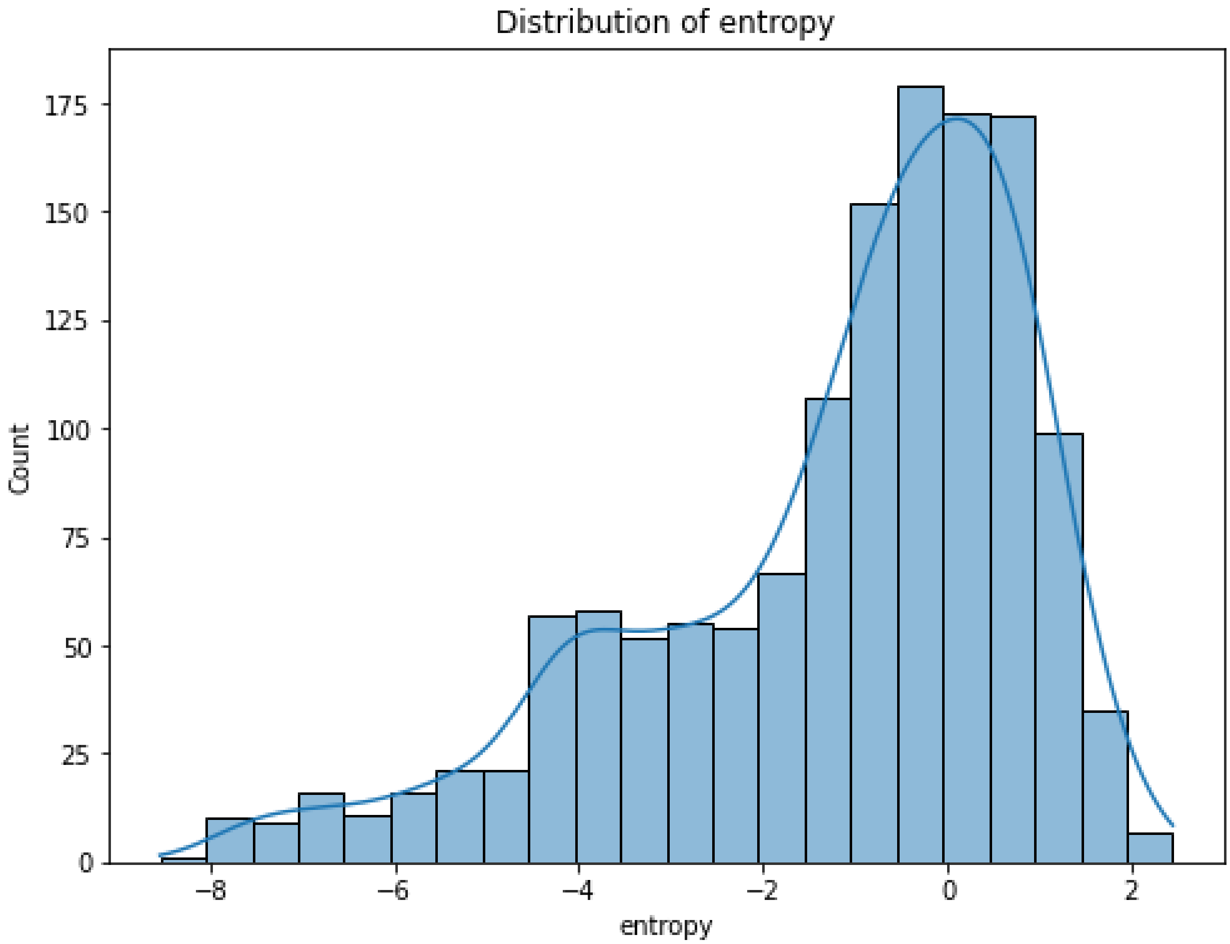

```
# 2. Distribusi setiap fitur kategorikal
def categorical_distribution(df):
    categorical_features =
df.select_dtypes(include=['object']).columns
    for feature in categorical_features:
        plt.figure(figsize=(8, 6))
        sns.countplot(x=feature, data=df)
        plt.title(f'Distribusi {feature}')
        plt.show()

def numerical_distribution(df):
    numeric_features = df.select_dtypes(include=['int64',
'float64']).columns
    for feature in numeric_features:
        plt.figure(figsize=(8, 6))
        sns.histplot(data=df, x=feature, kde=True)
```

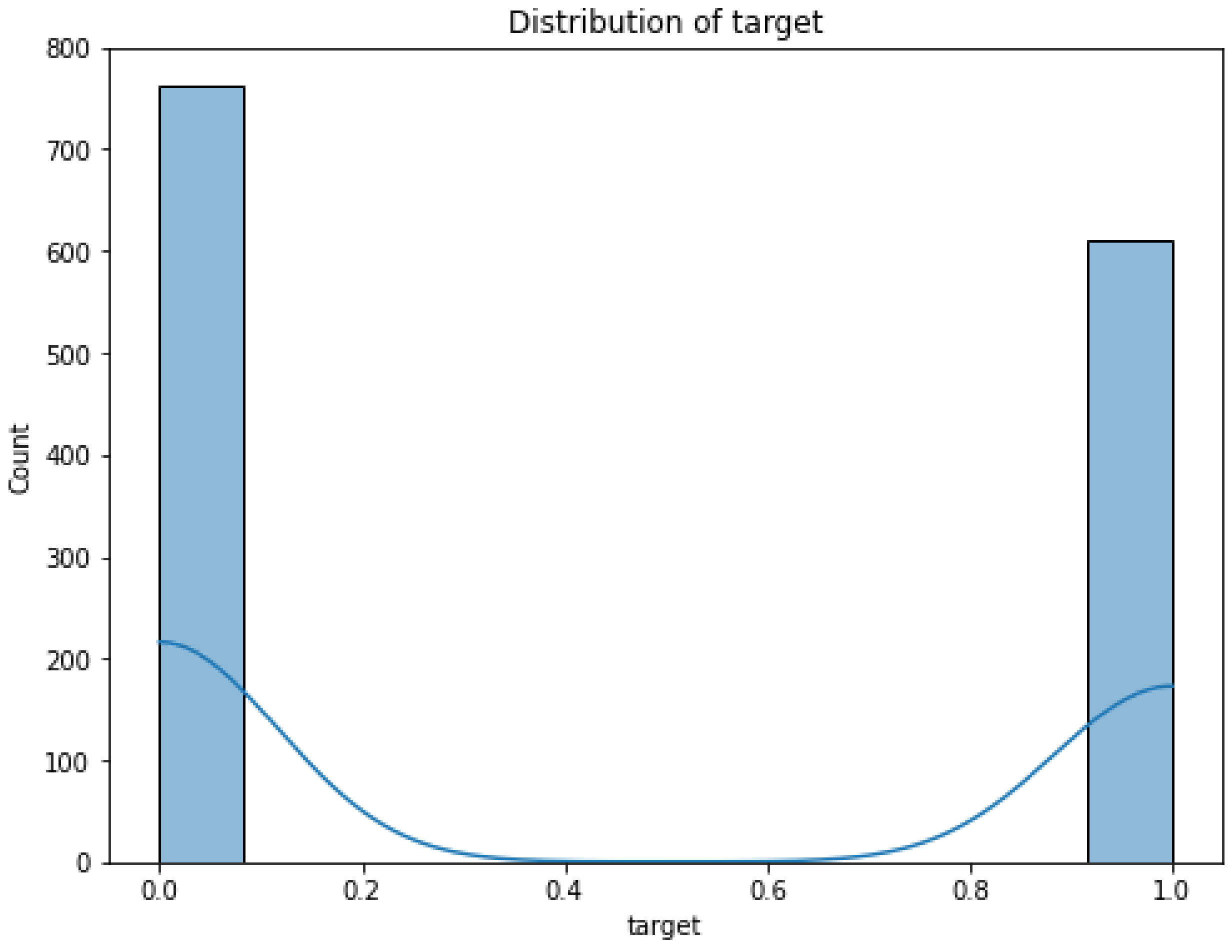
```
        plt.title(f'Distribution of {feature}')
        plt.show()

categorical_distribution(df)

numerical_distribution(df)
```




Distribution of skewness
Count
skewness
0
25
50
75
100
125
150
175
−15
−10
−5
0
5
10

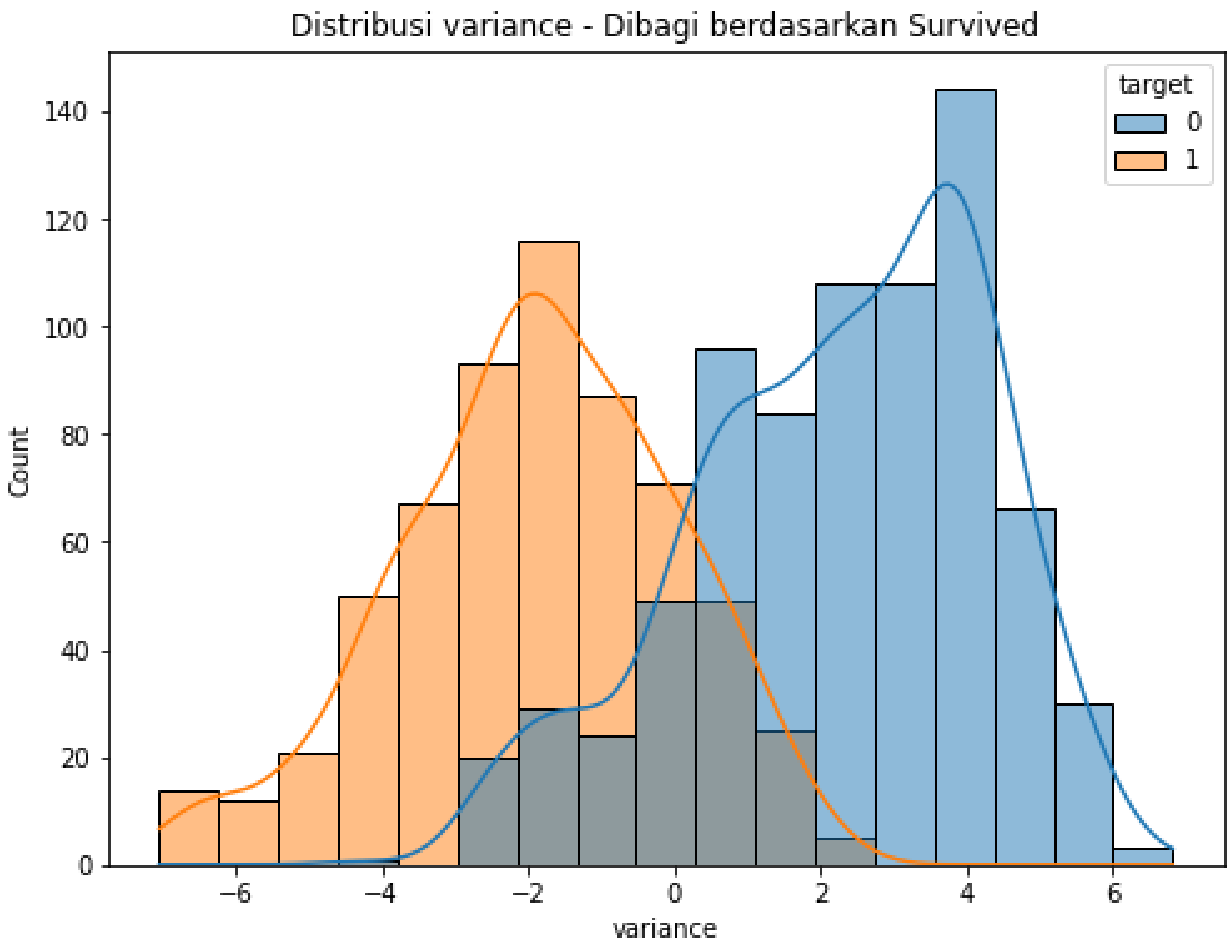

Distribution of curtosis
Count
curtosis
0
20
40
60
80
100
120
140
−5
0
5
10
15

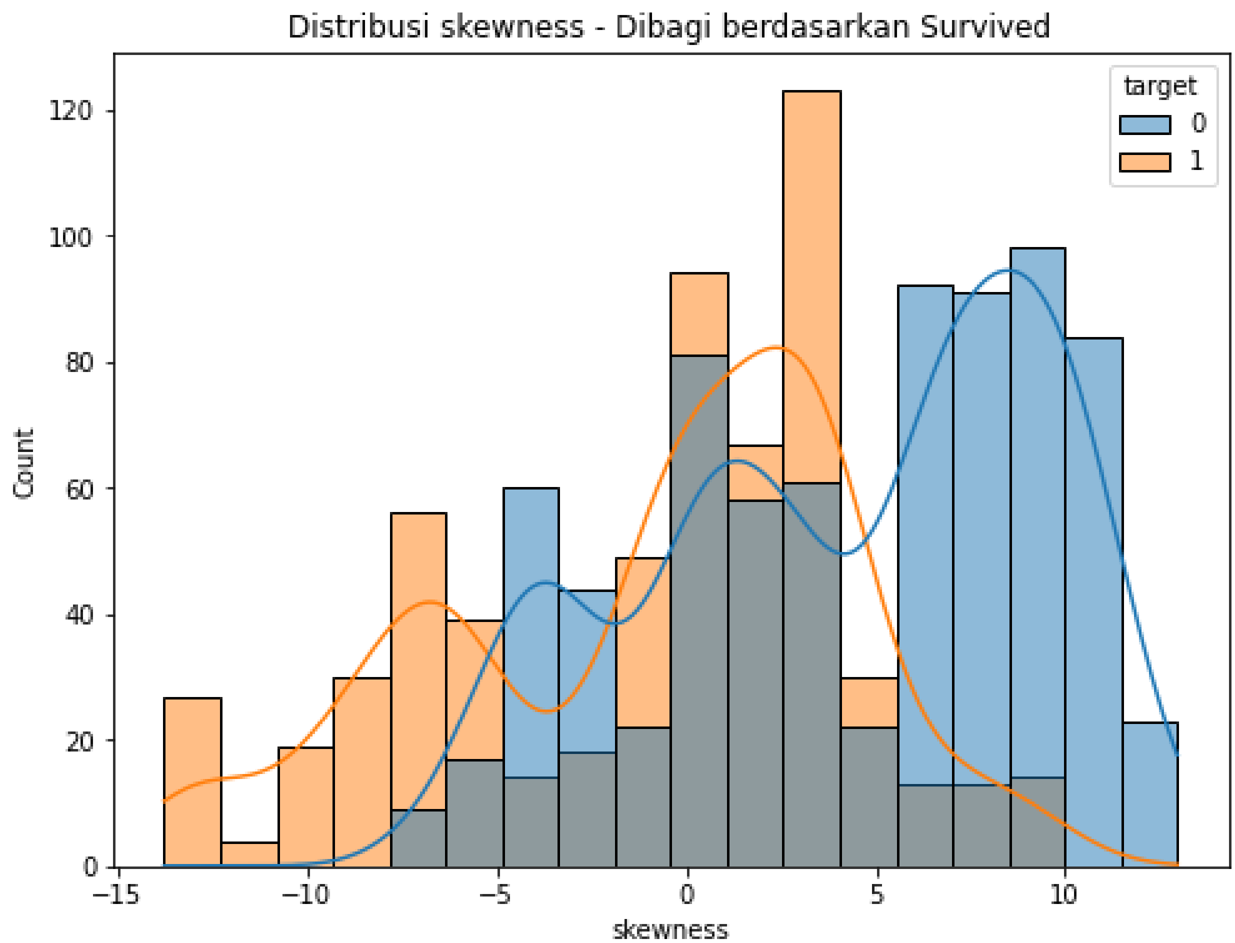

Distribution of entropy
Count
entropy
0
25
50
75
100
125
150
175
−8
−6
−4
−2
0
2

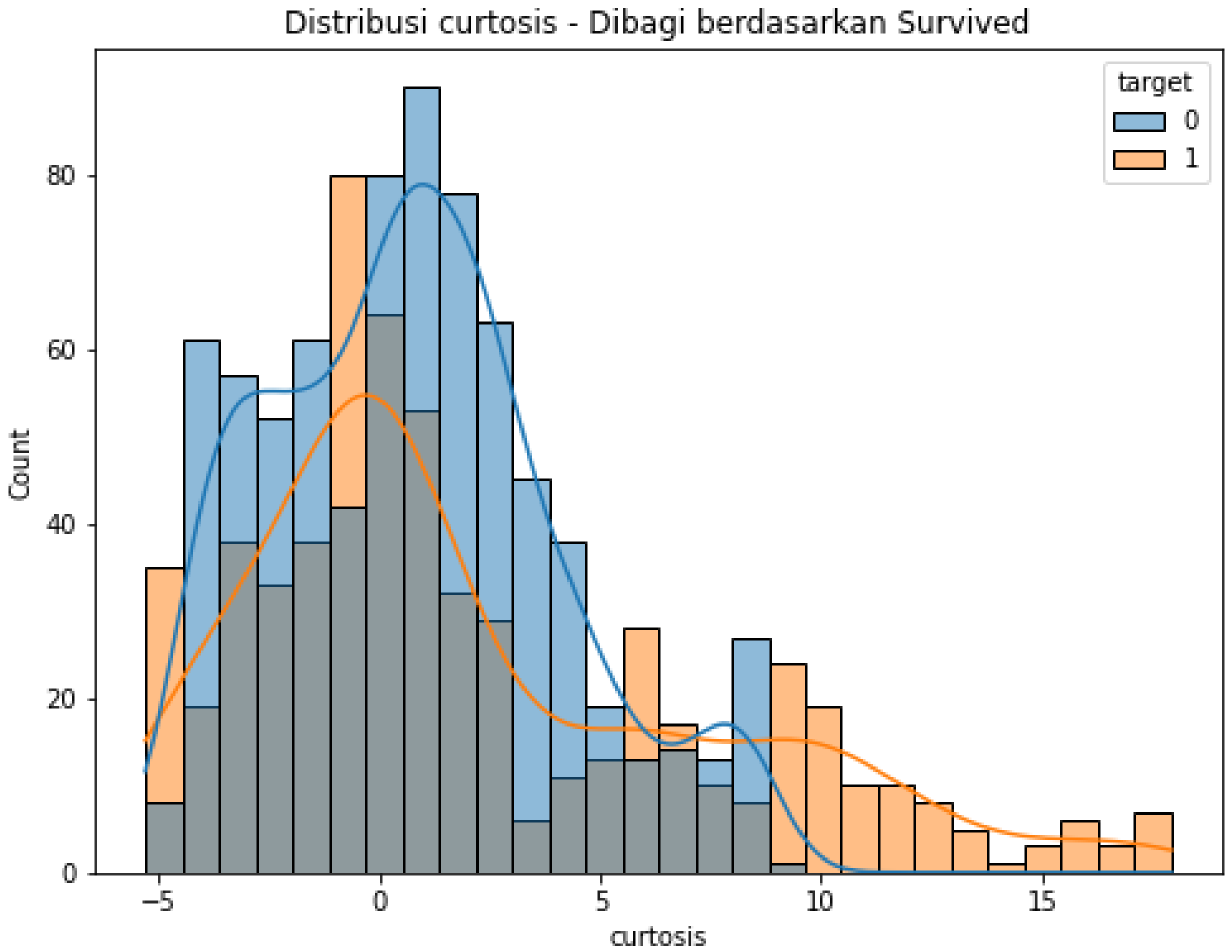



```
# 3. Visualisasikan distribusi dari tiap fitur, dengan data dibagi
tiap unique values fitur survived
def numerical_distribution_survived(df):
    numeric_features = df.select_dtypes(include=['int64',
'float64']).columns
    for feature in numeric_features:
        plt.figure(figsize=(8, 6))
        sns.histplot(data=df, x=feature, hue='target', kde=True)
        plt.title(f'Distribusi {feature} - Dibagi berdasarkan
Survived')
        plt.show()

def categorical_distribution_survived(df):
    categorical_features =
df.select_dtypes(include=['object']).columns
    for feature in categorical_features:
        plt.figure(figsize=(8, 6))
        sns.countplot(data=df, x=feature, hue='target')
        plt.title(f'Distribusi {feature} - Dibagi berdasarkan
```

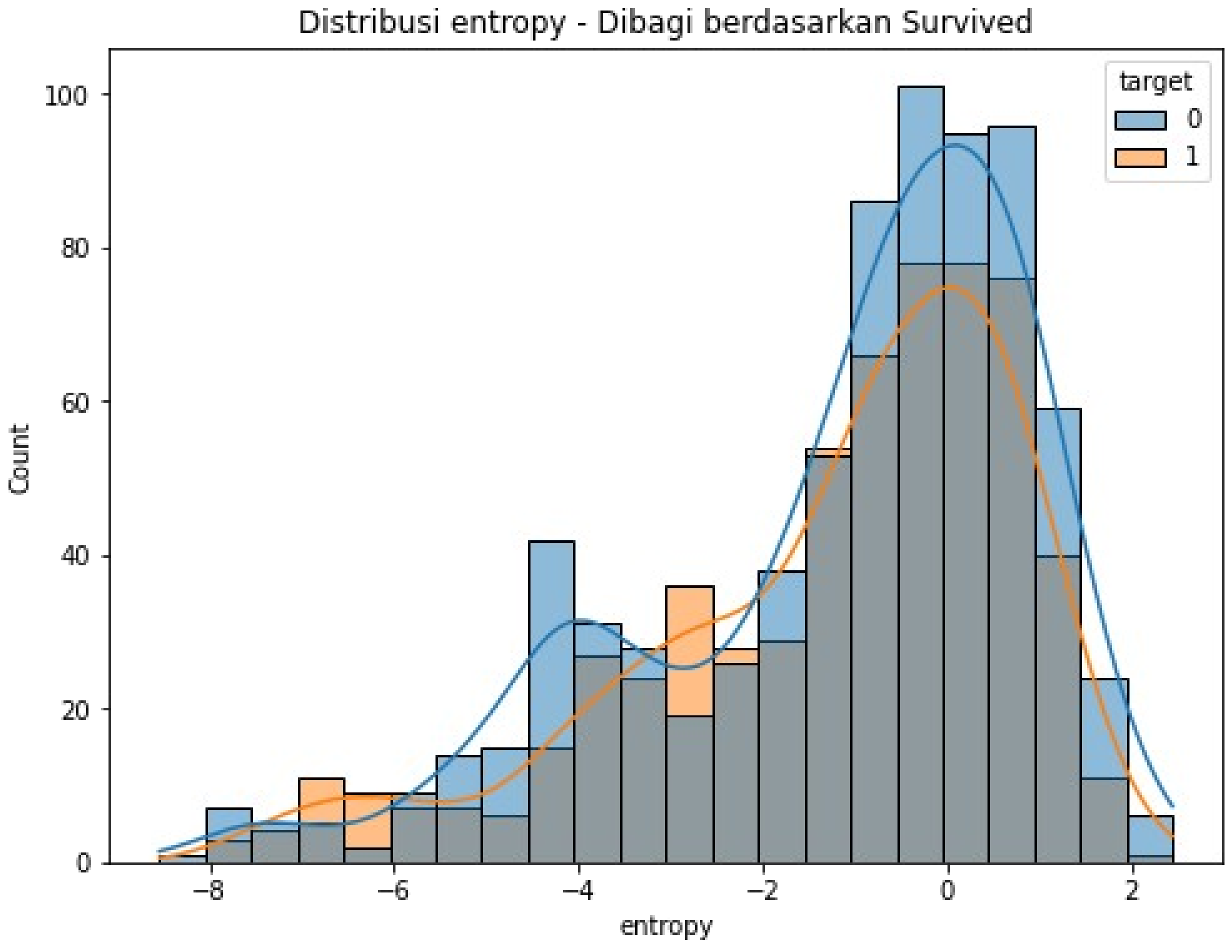
```
Survived')
        plt.show()

categorical_distribution_survived(df)

numerical_distribution_survived(df)
```

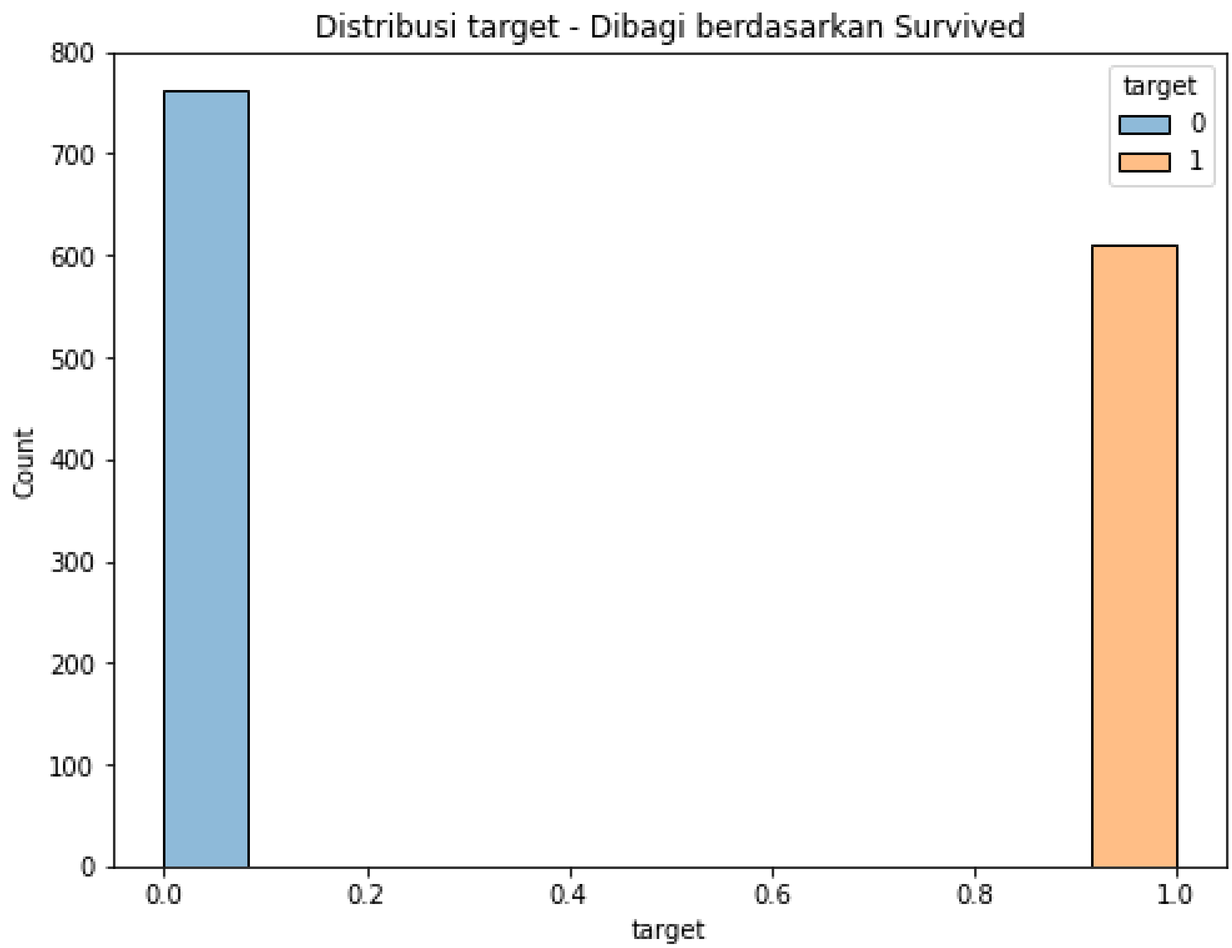



Distribusi skewness - Dibagi berdasarkan Survived
target
0
1
Count
skewness
120
100
80
60
40
20
0
−15
−10
−5
0
5
10



Distribusi curtosis - Dibagi berdasarkan Survived
target
0
1
Count
0
20
40
60
80
−5
0
5
10
15
curtosis


Distribusi entropy - Dibagi berdasarkan Survived
target
0
1
Count
100
80
60
40
20
0
−8
−6
−4
−2
0
2
entropy




## I.4

Lakukanlah analisa pada data lebih lanjut jika dibutuhkan, kemudian lakukanlah:

1. Penambahan fitur jika memungkinkan
2. Pembuangan fitur yang menurut kalian tidak dibutuhkan
3. Penanganan missing values
4. Transformasi data kategorikal menjadi numerikal (encoding), dengan metode yang kalian inginkan
5. Lakukan scaling dengan MinMaxScaler

Tidak ada missing values sehingga tidak perlu dilakukan penanganan

Tidak ada fitur kategorikal sehingga tidak perlu dilakukan encoding

Korelasi antar fitur terlihat sangat tinggi sehingga performa model machine learning bisa saja turun jika ada fitur didrop

## Handling Duplicate Rows

```
# Get the number of rows before removing duplicates
num_rows_before = len(df)

# Remove duplicate rows from df
df = df.drop_duplicates()

# Get the number of rows after removing duplicates
num_rows_after = len(df)

# Print the number of rows before and after removing duplicates
print(f"Number of rows before removing duplicates: {num_rows_before}")
print(f"Number of rows after removing duplicates: {num_rows_after}")
```

```
Number of rows before removing duplicates: 1372
Number of rows after removing duplicates: 1348
```

## Handling Outliers

```
df = find_outliers(df)
```

```
Outliers dari tiap fitur (metode IQR - Interquartile Range):
variance:
Tidak ada

skewness:
Tidak ada

curtosis:
      variance  skewness  curtosis   entropy  target
765    -3.8483  -12.8047   15.6824 -1.281000       1
780    -3.5801  -12.9309   13.1779 -2.567700       1
815    -3.1128   -6.8410   10.7402 -1.017200       1
816    -4.8554   -5.9037   10.9818 -0.821990       1
820    -4.0025  -13.4979   17.6772 -3.320200       1
821    -4.0173   -8.3123   12.4547 -1.437500       1
826    -4.2110  -12.4736   14.9704 -1.388400       1
841    -3.8858  -12.8461   12.7957 -3.135300       1
877    -5.1216   -5.3118   10.3846 -1.061200       1
881    -4.4861  -13.2889   17.3087 -3.219400       1
882    -4.3876   -7.7267   11.9655 -1.454300       1
887    -3.2692  -12.7406   15.5573 -0.141820       1
902    -2.8957  -12.0205   11.9149 -2.755200       1
937    -2.9020   -7.6563   11.8318 -0.842680       1
938    -4.3773   -5.5167   10.9390 -0.408200       1
942    -3.3793  -13.7731   17.9274 -2.032300       1
943    -3.1273   -7.1121   11.3897 -0.083634       1
948    -3.4917  -12.1736   14.3689 -0.616390       1
949    -3.1158   -8.6289   10.4403  0.971530       1
963    -3.3863  -12.9889   13.0545 -2.720200       1
```

```
998    -3.0866   -6.6362   10.5405 -0.891820       1
999    -4.7331   -6.1789   11.3880 -1.074100       1
1003   -3.8203  -13.0551   16.9583 -2.305200       1
1004   -3.7181   -8.5089   12.3630 -0.955180       1
1009   -3.5713  -12.4922   14.8881 -0.470270       1
1024   -3.0061  -12.2377   11.9552 -2.160300       1
1059   -3.2305   -7.2135   11.6433 -0.946130       1
1060   -4.8426   -4.9932   10.4052 -0.531040       1
1064   -3.6961  -13.6779   17.5795 -2.618100       1
1065   -3.6012   -6.5389   10.5234 -0.489670       1
1070   -3.1423  -13.0365   15.6773 -0.661650       1
1085   -2.6649  -12.8130   12.6689 -1.908200       1
1120   -3.1875   -7.5756   11.8678 -0.578890       1
1121   -4.6765   -5.6636   10.9690 -0.334490       1
1125   -3.5985  -13.6593   17.6052 -2.492700       1
1126   -3.3582   -7.2404   11.4419 -0.571130       1
1131   -4.0214  -12.8006   15.6199 -0.956470       1
1132   -3.3884   -8.2150   10.3315  0.981870       1
1146   -3.7300  -12.9723   12.9817 -2.684000       1
1181   -3.5895   -6.5720   10.5251 -0.163810       1
1182   -5.0477   -5.8023   11.2440 -0.390100       1
1186   -4.2440  -13.0634   17.1116 -2.801700       1
1187   -4.0218   -8.3040   12.5550 -1.509900       1
1192   -4.4018  -12.9371   15.6559 -1.680600       1
1207   -3.7930  -12.7095   12.7957 -2.825000       1
1243   -5.0676   -5.1877   10.4266 -0.867250       1
1247   -4.4775  -13.0303   17.0834 -3.034500       1
1248   -4.1958   -8.1819   12.1291 -1.601700       1
1253   -4.5531  -12.5854   15.4417 -1.498300       1
1268   -3.9411  -12.8792   13.0597 -3.312500       1
1304   -5.2943   -5.1463   10.3332 -1.118100       1
1308   -4.6338  -12.7509   16.7166 -3.216800       1
1309   -4.2887   -7.8633   11.8387 -1.897800       1
1314   -3.5060  -12.5667   15.1606 -0.752160       1
1329   -2.9672  -13.2869   13.4727 -2.627100       1
1364   -2.8391   -6.6300   10.4849 -0.421130       1
1365   -4.5046   -5.8126   10.8867 -0.528460       1
1369   -3.7503  -13.4586   17.5932 -2.777100       1
1370   -3.5637   -8.3827   12.3930 -1.282300       1

entropy:
      variance  skewness  curtosis  entropy  target
41    -0.20620    9.2207 -3.704400  -6.8103       0
45    -0.78690    9.5663 -3.786700  -7.5034       0
59    -0.78289   11.3603 -0.376440  -7.0495       0
194   -2.34100   12.3784  0.704030  -7.5836       0
202   -0.78689    9.5663 -3.786700  -7.5034       0
291   -2.21530   11.9625  0.078538  -7.7853       0
341   -1.18040   11.5093  0.155650  -6.8194       0
```

```
394   -2.26230   12.1177  0.288460  -7.7581       0
465   -2.69890   12.1984  0.676610  -8.5482       0
529   -1.38850   12.5026  0.691180  -7.5487       0
543   -1.42170   11.6542 -0.057699  -7.1025       0
562   -2.46040   12.7302  0.917380  -7.6418       0
581   -1.96670   11.8052 -0.404720  -7.8719       0
606   -1.42750   11.8797  0.416130  -6.9978       0
740   -2.44730   12.6247  0.735730  -7.6612       0
776   -5.90340    6.5679  0.676610  -6.6797       1
791   -4.47790    7.3708 -0.312180  -6.7754       1
837   -6.28150    6.6651  0.525810  -7.0107       1
852   -4.88610    7.0542 -0.172520  -6.9590       1
898   -5.24060    6.6258 -0.199080  -6.8607       1
959   -6.39790    6.4479  1.083600  -6.6176       1
974   -5.03010    7.5032 -0.133960  -7.5034       1
1142  -6.57730    6.8017  0.854830  -7.5344       1
1157  -5.20490    7.2590  0.070827  -7.3004       1
1164  -6.33640    9.2848  0.014275  -6.7844       1
1203  -6.73870    6.9879  0.678330  -7.5887       1
1218  -5.44140    7.2363  0.109380  -7.5642       1
1225  -6.52350    9.6014 -0.253920  -6.9642       1
1264  -6.65100    6.7934  0.686040  -7.5887       1
1279  -5.30120    7.3915  0.029699  -7.3987       1
1286  -6.42470    9.5311  0.022844  -6.8517       1
1325  -5.52500    6.3258  0.897680  -6.6241       1

target:
Tidak ada
```

### Feature Scaling

```
# Feature Scaling
# Split X and Y
X = df.drop('target', axis=1)
y = df['target']

# Scaling
scaler = MinMaxScaler()
X = scaler.fit_transform(X)
```

# II. Experiments Design

Tujuan dari bagian ini adalah peserta dapat memahami cara melakukan eksperimen mencari metode terbaik dengan benar. Hal ini meliputi:

1. Pembuatan model
2. Proses validasi

3. Hyperparameter tuning

# II.1

Tentukanlah metrics yang akan digunakan pada eksperimen kali ini (dapat lebih dari 1 metric)

1. Akurasi (Accuracy): Akurasi mengukur seberapa banyak prediksi yang benar dibagi dengan jumlah total data.
2. Presisi (Precision): Presisi mengukur seberapa banyak prediksi positif yang benar dibandingkan dengan total prediksi positif yang dilakukan oleh model.
3. Recall (Sensitivitas atau True Positive Rate): Recall mengukur seberapa banyak prediksi positif yang benar dibandingkan dengan total data sebenarnya yang bernilai positif.
4. F1-Score: F1-score adalah harmonic mean dari presisi dan recall.

# II.2

Bagi data dengan perbandingan 0.8 untuk data train dan 0.2 untuk data validasi

## Split Training and Test Set

```
# Bagi data menjadi train set dan test set (misalnya 80% train, 20%
test)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

# II.3

Lakukanlah:

1. Prediksi dengan menggunakan model Logistic Regression sebagai *baseline*
2. Tampilkan evaluasi dari model yang dibangun dari metrics yang anda tentukan pada II.1
3. Tampilkan confusion matrix

## Logistic Regression

```
# 1. Define and train model
model = LogisticRegression(random_state=42)
model.fit(X_train, y_train)

LogisticRegression(random_state=42)
```

### Training Score

```
# 2. Make predictions using the trained model for training score
y_pred = model.predict(X_train)
```

```python
# 3. Evaluate the model using the specified metrics
accuracy = accuracy_score(y_train, y_pred)
precision = precision_score(y_train, y_pred)
recall = recall_score(y_train, y_pred)
f1 = f1_score(y_train, y_pred)

print("Evaluation Metrics:")
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1 Score: {f1:.2f}")
```

```
Evaluation Metrics:
Accuracy: 0.99
Precision: 0.98
Recall: 1.00
F1 Score: 0.99
```

```python
# Generate the classification report
class_report = classification_report(y_train, y_pred)

# Classification Report
print("Classification Report:")
print(class_report)
```

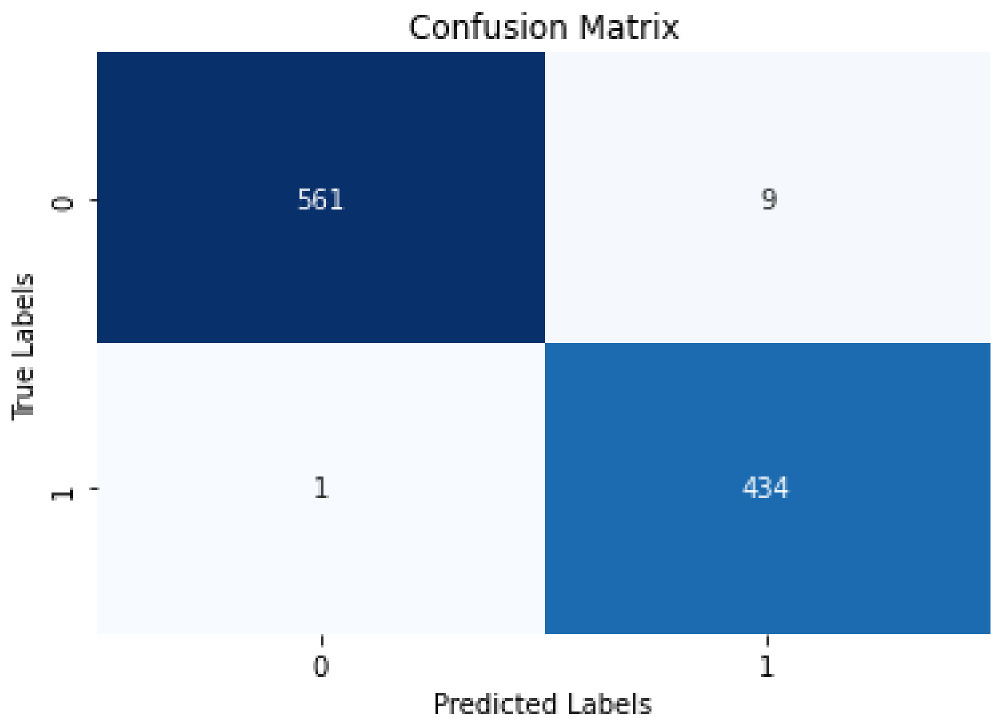
```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.98      0.99       570
           1       0.98      1.00      0.99       435

    accuracy                           0.99      1005
   macro avg       0.99      0.99      0.99      1005
weighted avg       0.99      0.99      0.99      1005
```

```python
# 4. Display the confusion matrix
conf_matrix = confusion_matrix(y_train, y_pred)
print("Confusion Matrix:")
print(conf_matrix)
```

```
Confusion Matrix:
[[561   9]
 [  1 434]]
```

```python
# Featmap for the confusion matrix
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues",
cbar=False)
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
```

```
plt.title("Confusion Matrix")
plt.show()
```




Test Score

```
# 2. Make predictions using the trained model for test score
y_pred = model.predict(X_test)

# 3. Evaluate the model using the specified metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("Evaluation Metrics:")
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1 Score: {f1:.2f}")

Evaluation Metrics:
Accuracy: 0.99
Precision: 0.98
Recall: 1.00
F1 Score: 0.99

# Generate the classification report
class_report_baseline = classification_report(y_test, y_pred)
```

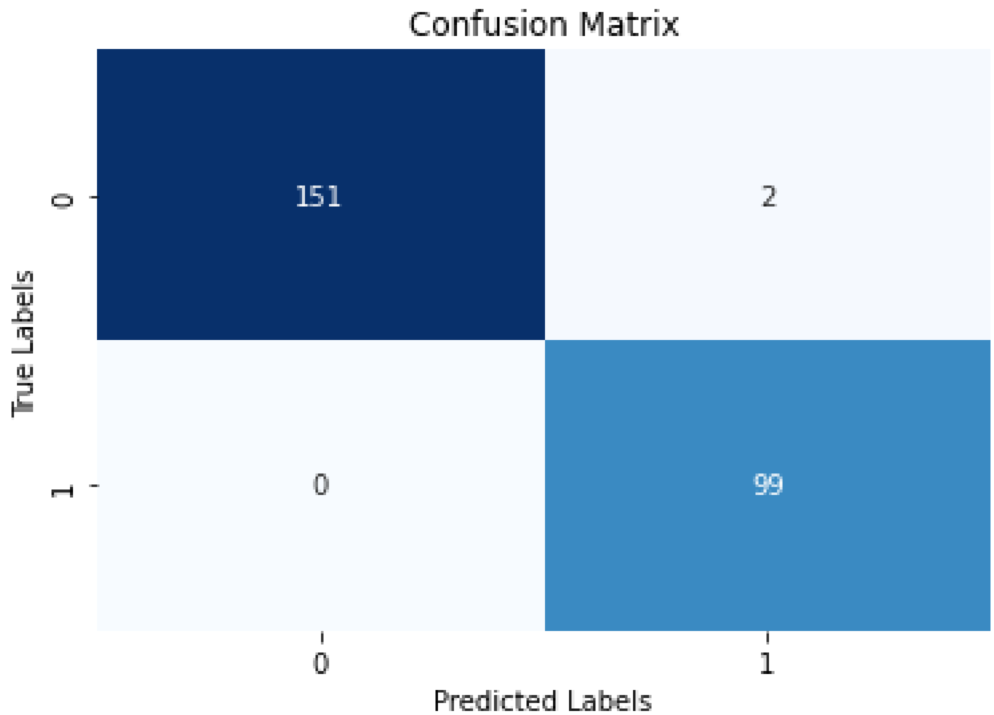
```
# Classification Report
print("Classification Report:")
print(class_report_baseline)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       153
           1       0.98      1.00      0.99        99

    accuracy                           0.99       252
   macro avg       0.99      0.99      0.99       252
weighted avg       0.99      0.99      0.99       252
```

```
# 4. Display the confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)
```

```
Confusion Matrix:
[[151   2]
 [  0  99]]
```

```
# Featmap for the confusion matrix
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues",
cbar=False)
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix")
plt.show()
```




# II.4

Lakukanlah:

1. Pembelajaran dengan model lain
2. Hyperparameter tuning model yang kalian pakai dengan menggunakan Grid Search (perhatikan random factor pada beberapa algoritma model)
3. Lakukan validasi dengan menggunakan cross validation

## Random Forest Classifier

```
# Define and train model
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)

RandomForestClassifier(random_state=42)

# Define hyperparameters
param_grid = {
    'n_estimators': [50, 100, 150],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Hyperparameter Tuning
grid_search = GridSearchCV(rf_model, param_grid, cv=5, n_jobs=-1)
grid_search.fit(X_train, y_train)
```

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(random_state=42),
n_jobs=-1,
             param_grid={'max_depth': [None, 10, 20],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5, 10],
                         'n_estimators': [50, 100, 150]})
```

```
# Get the best model
best_rf_model = grid_search.best_estimator_

# Validation with Cross Validation
cv_scores = cross_val_score(best_rf_model, X_train, y_train, cv=5,
scoring='accuracy')

# Print the results
print("Cross Validation Scores:")
print(cv_scores)
```

```
Cross Validation Scores:
[0.99502488 0.99502488 1.         0.99502488 0.9800995 ]
```

```
print("Best Model Hyperparameters:")
print(grid_search.best_params_)
```

```
Best Model Hyperparameters:
{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 5,
'n_estimators': 50}
```

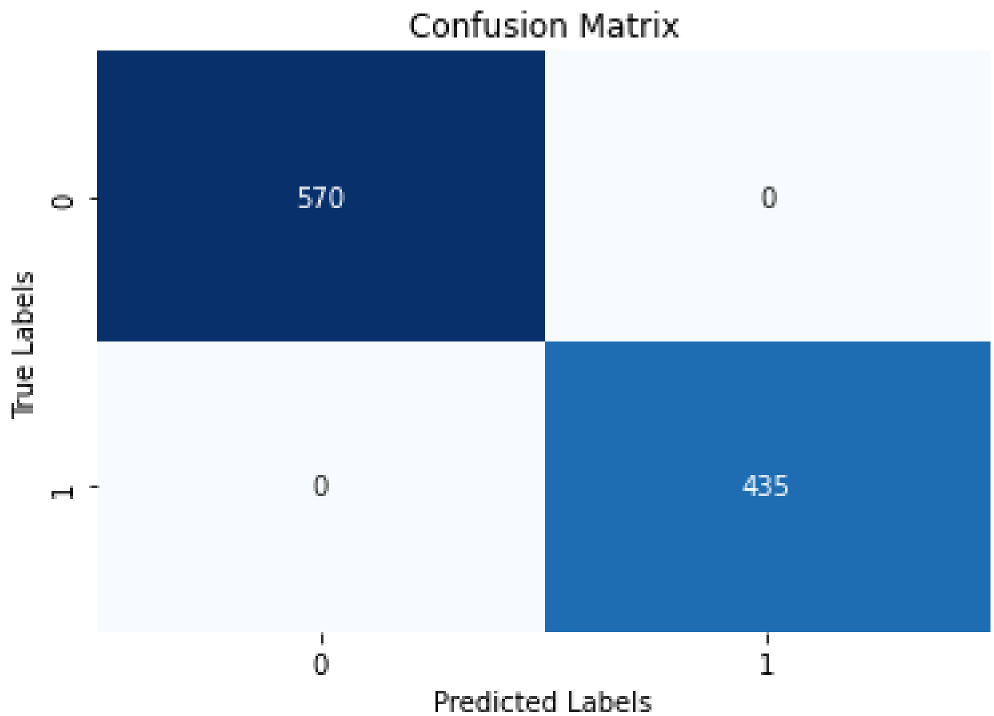
Training Score

```
# Evaluation metrics for the best model
y_pred = best_rf_model.predict(X_train)
class_report_rf = classification_report(y_train, y_pred)
conf_matrix = confusion_matrix(y_train, y_pred)

print("Classification Report:")
print(class_report_rf)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       570
           1       1.00      1.00      1.00       435

    accuracy                           1.00      1005
   macro avg       1.00      1.00      1.00      1005
weighted avg       1.00      1.00      1.00      1005
```

```
print("Confusion Matrix:")
print(conf_matrix)
```

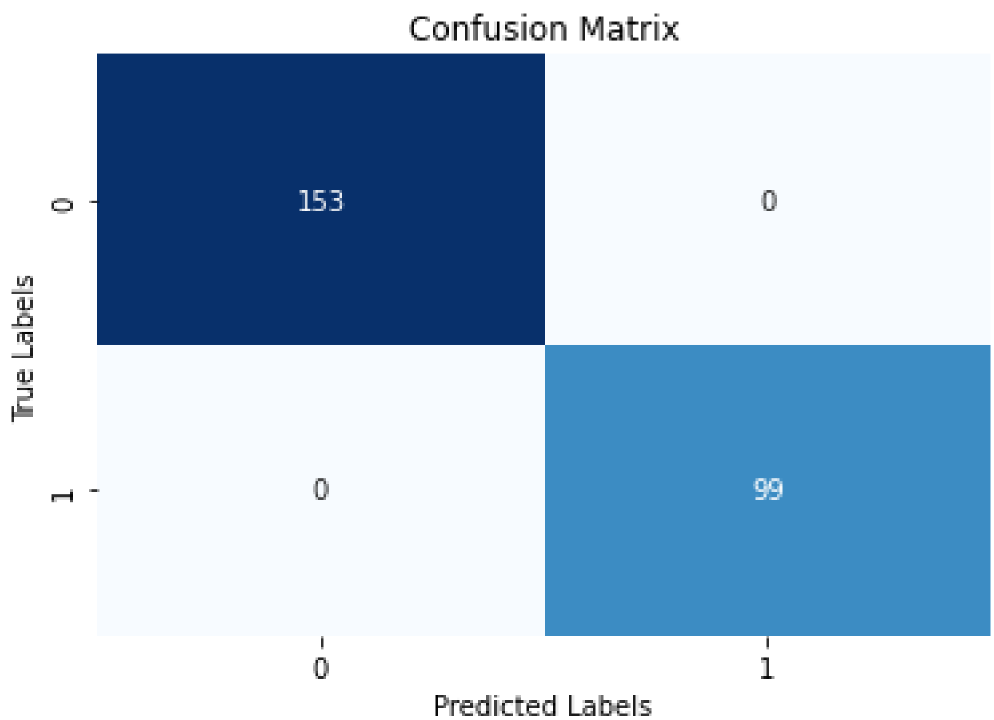
```
Confusion Matrix:
[[570   0]
 [  0 435]]

# Featmap for the confusion matrix
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues",
cbar=False)
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix")
plt.show()
```




Test Score

```
# Evaluation metrics for the best model
y_pred = best_rf_model.predict(X_test)
class_report_rf = classification_report(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print("Classification Report:")
print(class_report_rf)

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       153
           1       1.00      1.00      1.00        99
```

```
    accuracy                           1.00       252
   macro avg       1.00      1.00      1.00       252
weighted avg       1.00      1.00      1.00       252


print("Confusion Matrix:")
print(conf_matrix)

Confusion Matrix:
[[153   0]
 [  0  99]]

# Featmap for the confusion matrix
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues",
cbar=False)
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix")
plt.show()
```




# III. Improvement

Terdapat beberapa metode untuk melakukan peningkatan performa, contohnya adalah:

1. Melakukan oversampling / undersampling pada data
2. Menggabungkan beberapa model

Pada bagian ini, kalian diharapkan dapat:

1. Melakukan training dengan data hasil oversampling / undersampling dan melakukan validasi dengan benar
2. Memahami beberapa metode untuk menggabungkan beberapa model

# III.1

Lakukanlah:

1. Oversampling pada kelas minoritas pada data train, kemudian train dengan model *baseline* (II.3), lakukan validasi dengan data validasi. Data train dan validasi adalah data yang kalian bagi pada bagian II.2
2. Undersampling pada kelas mayoritas pada data train, kemudian train dengan model *baseline* (II.3) lakukan validasi dengan data validasi. Data train dan validasi adalah data yang kalian bagi pada bagian II.2

```
# III.1 Put your code here
# Oversampling on the minority class
oversampler = SMOTE(random_state=42)
X_train_oversampled, y_train_oversampled =
oversampler.fit_resample(X_train, y_train)

# Train with Logistic Regression (baseline model)
lr_model = LogisticRegression(random_state=42)
lr_model.fit(X_train_oversampled, y_train_oversampled)
```

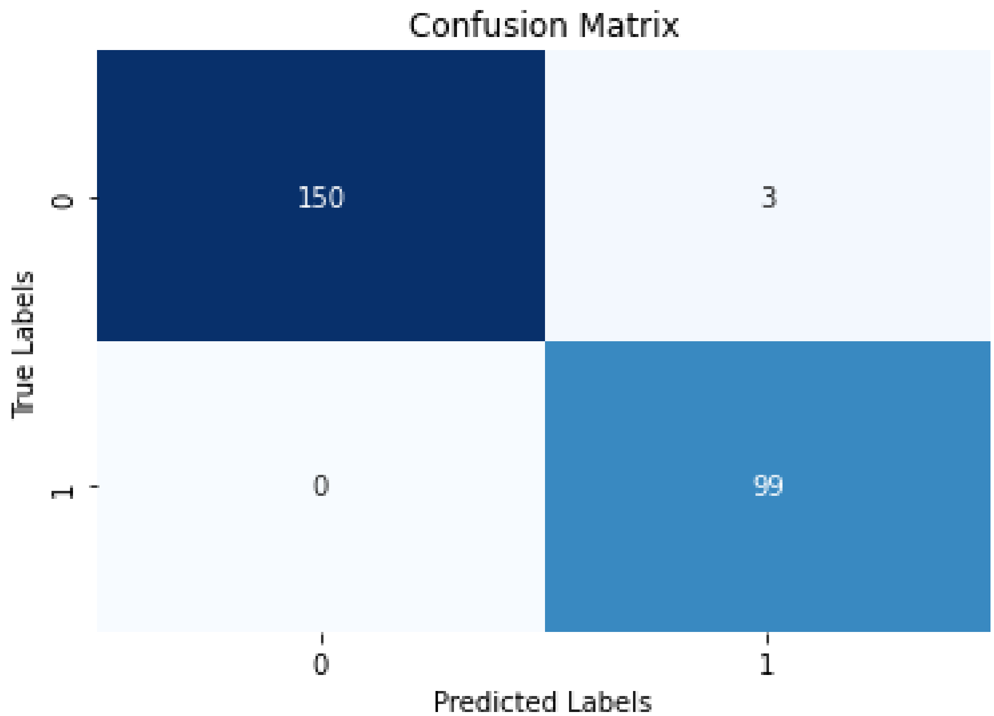
```
LogisticRegression(random_state=42)
```

```
# Validation
y_pred_lr = lr_model.predict(X_test)
class_report_oversampled = classification_report(y_test, y_pred_lr)
conf_matrix_lr = confusion_matrix(y_test, y_pred_lr)

print(class_report_oversampled)
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99       153
           1       0.97      1.00      0.99        99

    accuracy                           0.99       252
   macro avg       0.99      0.99      0.99       252
weighted avg       0.99      0.99      0.99       252
```

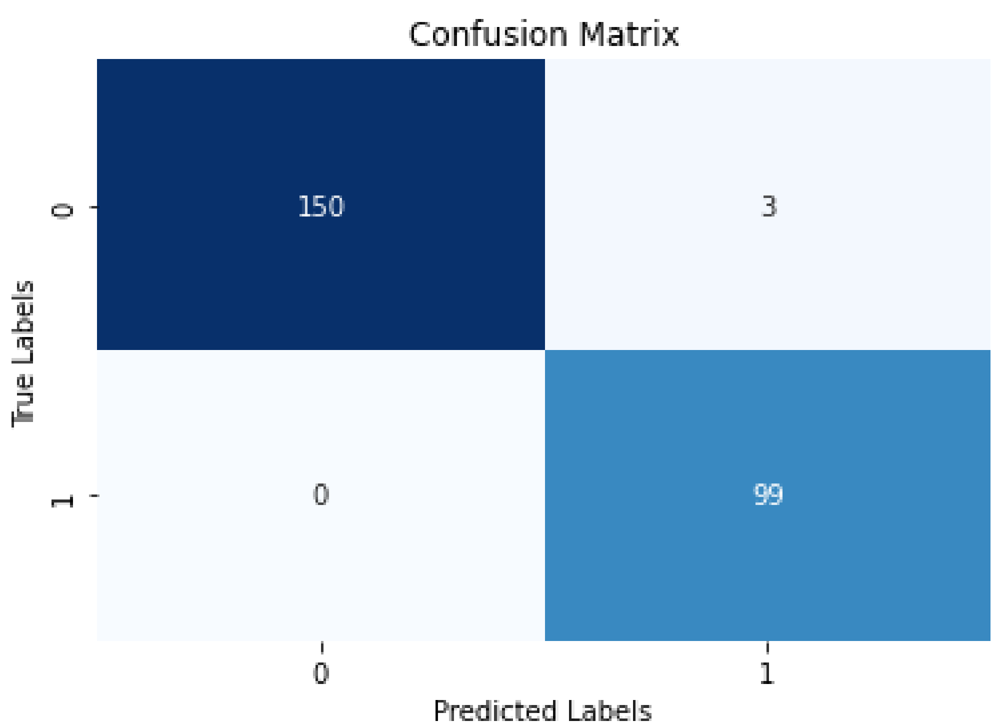
```
print(conf_matrix_lr)
```

```
[[150   3]
 [  0  99]]

# Featmap for the confusion matrix
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix_lr, annot=True, fmt="d", cmap="Blues",
cbar=False)
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix")
plt.show()
```




```
# Undersampling on the majority class
undersampler = RandomUnderSampler(random_state=42)
X_train_undersampled, y_train_undersampled =
undersampler.fit_resample(X_train, y_train)

# Train with Logistic Regression (baseline model)
lr_model_undersampled = LogisticRegression(random_state=42)
lr_model_undersampled.fit(X_train_undersampled, y_train_undersampled)

LogisticRegression(random_state=42)

# Validation
y_pred_lr_undersampled = lr_model_undersampled.predict(X_test)
class_report_lr_undersampled = classification_report(y_test,
y_pred_lr_undersampled)
```

```
conf_matrix_lr_undersampled = confusion_matrix(y_test,
y_pred_lr_undersampled)

print(class_report_lr_undersampled)

              precision    recall  f1-score   support

           0       1.00      0.98      0.99       153
           1       0.97      1.00      0.99        99

    accuracy                           0.99       252
   macro avg       0.99      0.99      0.99       252
weighted avg       0.99      0.99      0.99       252


print(conf_matrix_lr_undersampled)

[[150   3]
 [  0  99]]

# Featmap for the confusion matrix
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix_lr_undersampled, annot=True, fmt="d",
cmap="Blues", cbar=False)
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix")
plt.show()
```



# III.2

Lakukanlah:

1. Eksplorasi soft voting, hard voting, dan stacking
2. Buatlah model Logistic Regression dan SVM (boleh menggunakan model dengan beberapa parameter yang berbeda)
3. Lakukanlah soft voting dari model-model yang sudah kalian buat pada poin 2
4. Lakukan hard voting dari model-model yang sudah kalian buat pada poin 2
5. Lakukanlah stacking dengan final classifier adalah Logistic Regression dari model-model yang sudah kalian buat pada poin 2
6. Lakukan validasi dengan metrics yang kalian tentukan untuk poin 3, 4, dan 5

Put your answer for section III.2 point 1 here

```
# Define models
lr_model1 = LogisticRegression(C=1.0, random_state=42)
lr_model2 = LogisticRegression(C=0.1, random_state=42)
svm_model1 = SVC(kernel='linear', C=1.0, random_state=42,
probability=True)
svm_model2 = SVC(kernel='rbf', C=1.0, gamma='scale', random_state=42,
probability=True)

# Soft Voting
soft_voting_clf = VotingClassifier(estimators=[('lr1', lr_model1),
('lr2', lr_model2), ('svm1', svm_model1), ('svm2', svm_model2)],
voting='soft')
soft_voting_clf.fit(X_train, y_train)
y_pred_soft_voting = soft_voting_clf.predict(X_test)

# Evaluation Soft Voting
class_report_softvote = classification_report(y_test,
y_pred_soft_voting)
print(class_report_softvote)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       153
           1       1.00      1.00      1.00        99

    accuracy                           1.00       252
   macro avg       1.00      1.00      1.00       252
weighted avg       1.00      1.00      1.00       252
```

```
# Hard voting
hard_voting_clf = VotingClassifier(estimators=[('lr1', lr_model1),
('lr2', lr_model2), ('svm1', svm_model1), ('svm2', svm_model2)],
voting='hard')
```

```
hard_voting_clf.fit(X_train, y_train)
y_pred_hard_voting = hard_voting_clf.predict(X_test)

# Evaluation Hard Voting
class_report_hardvote = classification_report(y_test,
y_pred_hard_voting)
print(class_report_hardvote)

              precision    recall  f1-score   support

           0       1.00      0.99      0.99       153
           1       0.98      1.00      0.99        99

    accuracy                           0.99       252
   macro avg       0.99      0.99      0.99       252
weighted avg       0.99      0.99      0.99       252


# Stacking with Logistic Regression as the final classifier
estimators = [('lr1', lr_model1), ('lr2', lr_model2), ('svm1',
svm_model1), ('svm2', svm_model2)]
stacking_clf = StackingClassifier(estimators=estimators,
final_estimator=LogisticRegression())
stacking_clf.fit(X_train, y_train)
y_pred_stacking = stacking_clf.predict(X_test)

# Evaluation Stacking
class_report_stacking = classification_report(y_test, y_pred_stacking)
print(class_report_stacking)

              precision    recall  f1-score   support

           0       1.00      1.00      1.00       153
           1       1.00      1.00      1.00        99

    accuracy                           1.00       252
   macro avg       1.00      1.00      1.00       252
weighted avg       1.00      1.00      1.00       252
```

# IV. Analisis

Bandingkan hasil dari:

1. Model Baseline (II.3)
2. Model lain (II.4)
3. Hasil undersampling
4. Hasil oversampling
5. Hasil soft voting

6. Hasil hard voting
7. Hasil stacking

```
# 1. Model Baseline
print(class_report_baseline)

              precision    recall  f1-score   support

           0       1.00      0.99      0.99       153
           1       0.98      1.00      0.99        99

    accuracy                           0.99       252
   macro avg       0.99      0.99      0.99       252
weighted avg       0.99      0.99      0.99       252


# 2. Model RFC
print(class_report_rf)

              precision    recall  f1-score   support

           0       1.00      1.00      1.00       153
           1       1.00      1.00      1.00        99

    accuracy                           1.00       252
   macro avg       1.00      1.00      1.00       252
weighted avg       1.00      1.00      1.00       252


# 3. Model Undersampled
print(class_report_lr_undersampled)

              precision    recall  f1-score   support

           0       1.00      0.98      0.99       153
           1       0.97      1.00      0.99        99

    accuracy                           0.99       252
   macro avg       0.99      0.99      0.99       252
weighted avg       0.99      0.99      0.99       252


# 4. Model Oversampled
print(class_report_oversampled)

              precision    recall  f1-score   support

           0       1.00      0.98      0.99       153
           1       0.97      1.00      0.99        99

    accuracy                           0.99       252
   macro avg       0.99      0.99      0.99       252
```

```
weighted avg       0.99      0.99      0.99       252

# 5. Model Soft Voting
print(class_report_softvote)

              precision    recall  f1-score   support

           0       1.00      1.00      1.00       153
           1       1.00      1.00      1.00        99

    accuracy                           1.00       252
   macro avg       1.00      1.00      1.00       252
weighted avg       1.00      1.00      1.00       252

# 6. Model Hard Vote
print(class_report_hardvote)

              precision    recall  f1-score   support

           0       1.00      0.99      0.99       153
           1       0.98      1.00      0.99        99

    accuracy                           0.99       252
   macro avg       0.99      0.99      0.99       252
weighted avg       0.99      0.99      0.99       252

# 7. Model Stacking
print(class_report_stacking)

              precision    recall  f1-score   support

           0       1.00      1.00      1.00       153
           1       1.00      1.00      1.00        99

    accuracy                           1.00       252
   macro avg       1.00      1.00      1.00       252
weighted avg       1.00      1.00      1.00       252
```

Untuk analisis, berdasarkan dari evaluasi metrik-metrik di atas, terlihat bahwa seluruh model menghasilkan performa yang sangat baik karena untuk semua test validasi, semua nilai metrik di atas 95% sehingga untuk dataset yang dipakai dalam eksperimen ini sangat bagus, membuat algoritma machine learning berhasil menghasilkan performa yang baik. Maka, terlihat bahwa perbedaan nilai metrik-metrik tidak jauh berbeda. Namun, secara teori, memang model yang lebih engineered seperti softvote, hardvote, dan stacking seharusnya menghasilkan performa yang lebih baik dibandingkan baseline model. Balancing class juga diperlukan agar model tidak overfit ke suatu class, dengan teknik oversampling atau undersampling. Kedua teknik ada kekurangan dan kelebihannya baik dari segi jumlah data dan keorisinalitas data.