Sains Data

Fetal Cardiotocography

Metode *Decision Tree*, *Keras,* dan *Multilayer Perceptron* untuk masalah Klasifikasi *Fetal Cardiography.*

Project Presentation

Kelompok 4

Anggota Kelompok

01.

M. Alvero Johansyah

2106726106

02.

Jesica Andriana

2106726163

03.

Diyara Aulia

2106702131

Sains Data

List of Content

Sains Data

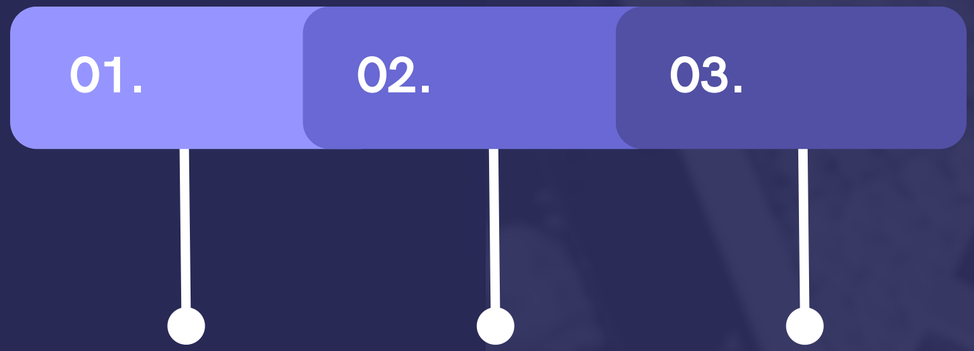
List of Content

Rumusan Masalah

Latar Belakang

Informasi yang diperoleh dari kardiotokografi, digunakan untuk mengidentifikasi awal keadaan patologis (gangguan pada janin, perkembangan penyakit pada janin atau hipoksia dan lain-lain)

Maka kita akan memanfaatkan Metode Klasifikasi dalam topik "*Fetal Cardiotocography*" yang nantinya akan di bedakan menjadi tiga bagian yaitu :



CTG.csv

	A	B	C	D	E	F
1	FileName	Date	SegFile	b	e	LBE
2	Varlab10.txt	12/1/1996	CTG0001.txt	240	357	120
3	Fmcs_1.txt	5/3/1996	CTG0002.txt	5	632	132
4	Fmcs_1.txt	5/3/1996	CTG0003.txt	177	779	133
5	Fmcs_1.txt	5/3/1996	CTG0004.txt	411	1192	134
6	Fmcs_1.txt	5/3/1996	CTG0005.txt	533	1147	132
7	Fmcs_2.txt	5/3/1996	CTG0006.txt	0	953	134
8	Fmcs_2.txt	5/3/1996	CTG0007.txt	240	953	134
9	Hasc_1.txt	2/22/1995	CTG0008.txt	62	679	122
10	Hasc_1.txt	2/22/1995	CTG0009.txt	120	779	122
11	Hasc_1.txt	2/22/1995	CTG0010.txt	181	1192	122
12	Hasc3.txt	2/22/1995	CTG0011.txt	0	1199	151
13	Hasc3.txt	2/22/1995	CTG0012.txt	57	1074	150
14	Mcslrc_1.txt	1/8/1995	CTG0013.txt	52	840	131
15	Mcslrc_1.txt	1/8/1995	CTG0014.txt	531	1192	131

Informasi DataSet

Kami telah mendapatkan DataSet melalui UCI terdapat sekitar

40 Fitur

2129 Baris

Sains Data

List of Content

Model yang dipakai

Keras

Keras adalah salah satu pengembangan API machine learning TensorFlow dengan fokus utama untuk menghasilkan komputasi yang cepat dibandingkan model lainnya.

Multilayer Perceptron

Multilayer perceptron (MLP) adalah jaringan saraf tiruan feedforward. MLP dicirikan oleh beberapa lapisan node input yang terhubung sebagai grafik.

Decision Tree

Decision tree adalah model machine learning yang menggambarkan hubungan antara fitur-fitur dan target dalam bentuk struktur pohon, di mana setiap simpul (node) mewakili keputusan berdasarkan fitur-fitur yang dipilih.

Model Data

Tahapan Proses Model Data

LANGKAH 1
Data Observation

LANGKAH 2
Data Cleaning

LANGKAH 3
EDA

LANGKAH 4
Data Preprocessing

LANGKAH 5
Model Selection

LANGKAH 6
Hyperparameter Tuning

LANGKAH 7
Model Evaluation

LANGKAH 8
Feature Importance

Google Collab

Shortcut untuk melihat hasil *code python* dengan Google Collab

1 Decision Tree

2 Keras

3 Multilayer Perception

Sains Data

List of Content

Tahapan Proses Model Data

LANGKAH 1

Data Observation

Decision Tree | Keras | Multilayer Perceptron

Data Observation

Import Dataset

```
[196] from google.colab import drive
      drive.mount('/content/drive')
      !pip install pandas
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (1.5.3)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2022.7.1)
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.22.4)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
```

```
[197] pd.set_option('display.max_columns', None)
      df = pd.read_csv("/content/drive/MyDrive/Project Sains Data/CTG.csv")
      df
```

	FileName	Date	SegFile	b	e	LBE	LB	AC	FM	UC	ASTV	MSTV	ALTV	MLTV	DL	DS	DP	DR	Width	Min	Max	Nmax	Nzeros	Mo
0	Variab10.txt	12/1/1996	CTG0001.txt	240.0	357.0	120.0	120.0	0.0	0.0	0.0	73.0	0.5	43.0	2.4	0.0	0.0	0.0	0.0	64.0	62.0	126.0	2.0	0.0	120
1	Fmcs_1.txt	5/3/1996	CTG0002.txt	5.0	632.0	132.0	132.0	4.0	0.0	4.0	17.0	2.1	0.0	10.4	2.0	0.0	0.0	0.0	130.0	68.0	198.0	6.0	1.0	141
2	Fmcs_1.txt	5/3/1996	CTG0003.txt	177.0	779.0	133.0	133.0	2.0	0.0	5.0	16.0	2.1	0.0	13.4	2.0	0.0	0.0	0.0	130.0	68.0	198.0	5.0	1.0	141
3	Fmcs_1.txt	5/3/1996	CTG0004.txt	411.0	1192.0	134.0	134.0	2.0	0.0	6.0	16.0	2.4	0.0	23.0	2.0	0.0	0.0	0.0	117.0	53.0	170.0	11.0	0.0	137
4	Fmcs_1.txt	5/3/1996	CTG0005.txt	533.0	1147.0	132.0	132.0	4.0	0.0	5.0	16.0	2.4	0.0	19.9	0.0	0.0	0.0	0.0	117.0	53.0	170.0	9.0	0.0	137
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2124	S8001045.dsp	6/6/1998	CTG2127.txt	1576.0	3049.0	140.0	140.0	1.0	0.0	9.0	78.0	0.4	27.0	7.0	0.0	0.0	0.0	0.0	66.0	103.0	169.0	6.0	0.0	152
2125	S8001045.dsp	6/6/1998	CTG2128.txt	2796.0	3415.0	142.0	142.0	1.0	1.0	5.0	74.0	0.4	36.0	5.0	0.0	0.0	0.0	0.0	42.0	117.0	159.0	2.0	1.0	145
2126	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
2127	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0	0.0	0.0	0.0	NaN	NaN	NaN	NaN	NaN	Na
2128	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	564.0	23.0	87.0	7.0	91.0	50.7	16.0	1.0	4.0	0.0	NaN	NaN	NaN	NaN	NaN	Na

0s completed at 8:03 PM

Fitur dan Ukuran Dataset

```
[199] df.columns
```

```
Index(['FileName', 'Date', 'SegFile', 'b', 'e', 'LBE', 'LB', 'AC', 'FM', 'UC',
       'ASTV', 'MSTV', 'ALTV', 'MLTV', 'DL', 'DS', 'DP', 'DR', 'Width', 'Min',
       'Max', 'Nmax', 'Nzeros', 'Mode', 'Mean', 'Median', 'Variance',
       'Tendency', 'A', 'B', 'C', 'D', 'E', 'AD', 'DE', 'LD', 'FS', 'SUSP',
       'CLASS', 'NSP'],
      dtype='object')
```

```
[200] df.shape
```

```
(2129, 40)
```

Data Observation

Describe Dataset

```
df.describe()
```

	b	e	LBE	LB	AC	FM	UC	ASTV	MSTV	ALTV	MLTV	DL	DS
count	2126.000000	2126.000000	2126.000000	2126.000000	2126.000000	2127.000000	2127.000000	2127.000000	2127.000000	2127.000000	2127.000000	2128.000000	2128.000000
mean	878.439793	1702.877234	133.303857	133.303857	2.722484	7.503056	3.669017	47.008933	1.335449	9.884814	8.207616	1.576128	0.003759
std	894.084748	930.919143	9.840844	9.840844	3.560850	39.030452	2.877148	17.210648	0.891543	18.476534	5.701926	2.517794	0.061213
min	0.000000	287.000000	106.000000	106.000000	0.000000	0.000000	0.000000	12.000000	0.200000	0.000000	0.000000	0.000000	0.000000
25%	55.000000	1009.000000	126.000000	126.000000	0.000000	0.000000	1.000000	32.000000	0.700000	0.000000	4.600000	0.000000	0.000000
50%	538.000000	1241.000000	133.000000	133.000000	1.000000	0.000000	3.000000	49.000000	1.200000	0.000000	7.400000	0.000000	0.000000
75%	1521.000000	2434.750000	140.000000	140.000000	4.000000	2.000000	5.000000	61.000000	1.700000	11.000000	10.800000	3.000000	0.000000
max	3296.000000	3599.000000	160.000000	160.000000	26.000000	564.000000	23.000000	87.000000	7.000000	91.000000	50.700000	16.000000	1.000000

Cek Kekosongan Dataset

```
+ Code + Text
df.isna().sum()
```

```
FileName    3
Date        3
SegFile     3
b           3
e           3
LBE         3
LB          3
AC          3
FM          3
UC          2
ASTV        2
MSTV        2
ALTV        2
MLTV        2
DL          1
DS          1
DP          1
DR          1
Width       3
Min         3
Max         3
Nmax        3
Nzeros      3
Mode        3
Mean        3
```

Cek Data Duplikat

```
[10] #Melihat jumlah data duplikat pada dataset
     df.duplicated().sum()
```

```
0
```

Tahapan Proses Model Data

LANGKAH 2

Data Cleaning

Decision Tree | Keras | Multilayer Perceptron

Data Cleaning

Data Cleaning

Terdapat Beberapa Fitur Kolom dan Baris yang perlu dilakukan cleaning

Penghapusan Fitur Kolom

Terdapat fitur seperti FileName, Date, SegFile yang tidak berpengaruh terhadap model dan terdapat juga beberapa fitur yang mempunyai banyak missing value hampir di semua indeks, sehingga akan di drop saja.

Penghapusan Fitur Baris

Perhatikan bahwa, sebelumnya telah diketahui terdapat 3 baris terakhir yang memiliki banyak missing value di hampir setiap kolom fitur, sehingga akan dilakukan drop value.

```
df = df.drop(columns=['FileName', 'Date', 'SegFile', 'b', 'e', 'A', 'B',
                      'C', 'D','E', 'AD', 'DE', 'LD', 'FS',
                      'SUSP', 'DR'], axis=1)
df
```

```
df = df.iloc[:2126]
```

Tahapan Proses Model Data

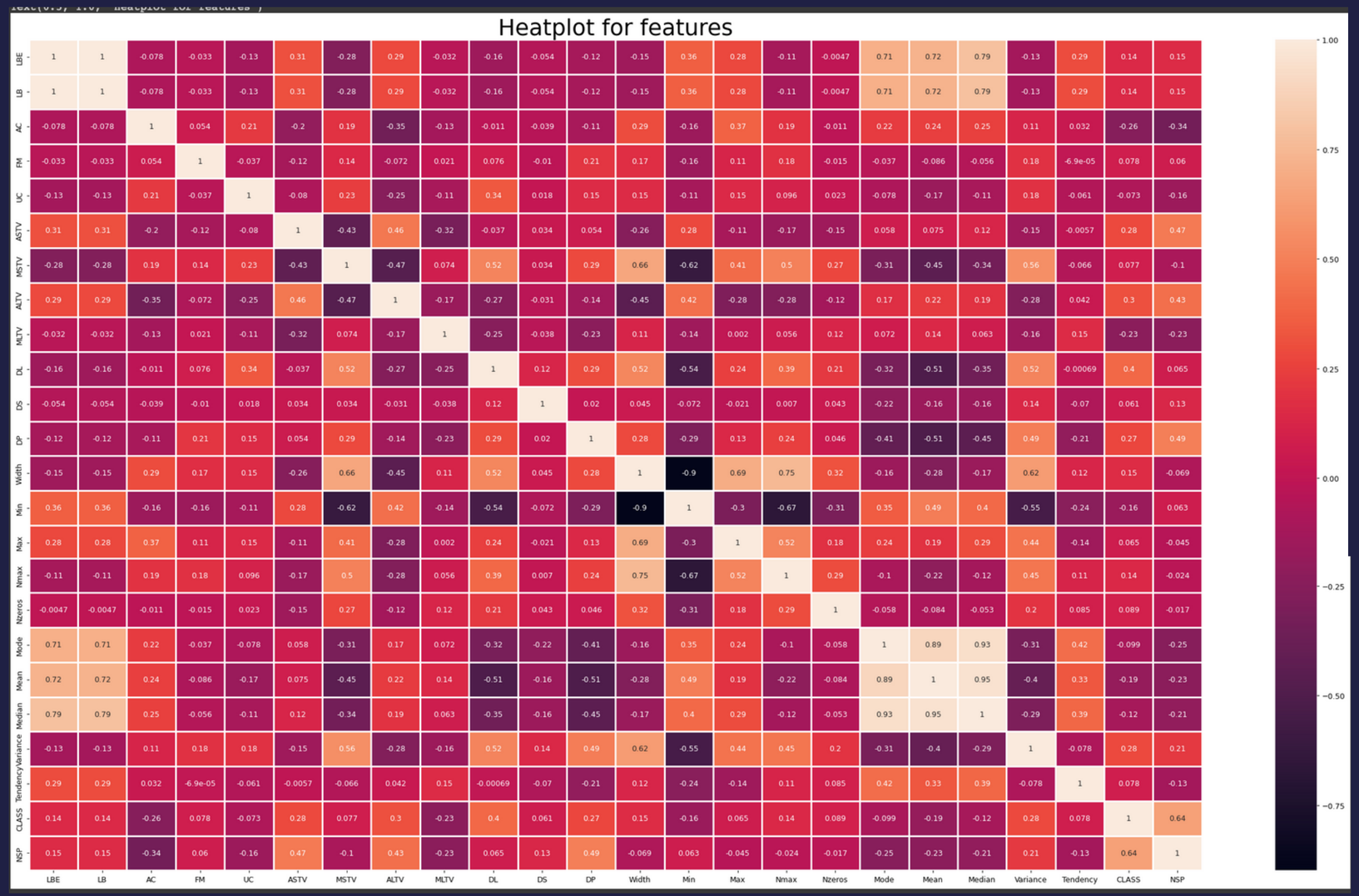
LANGKAH 3

EDA

Decision Tree | Keras | Multilayer Perceptron

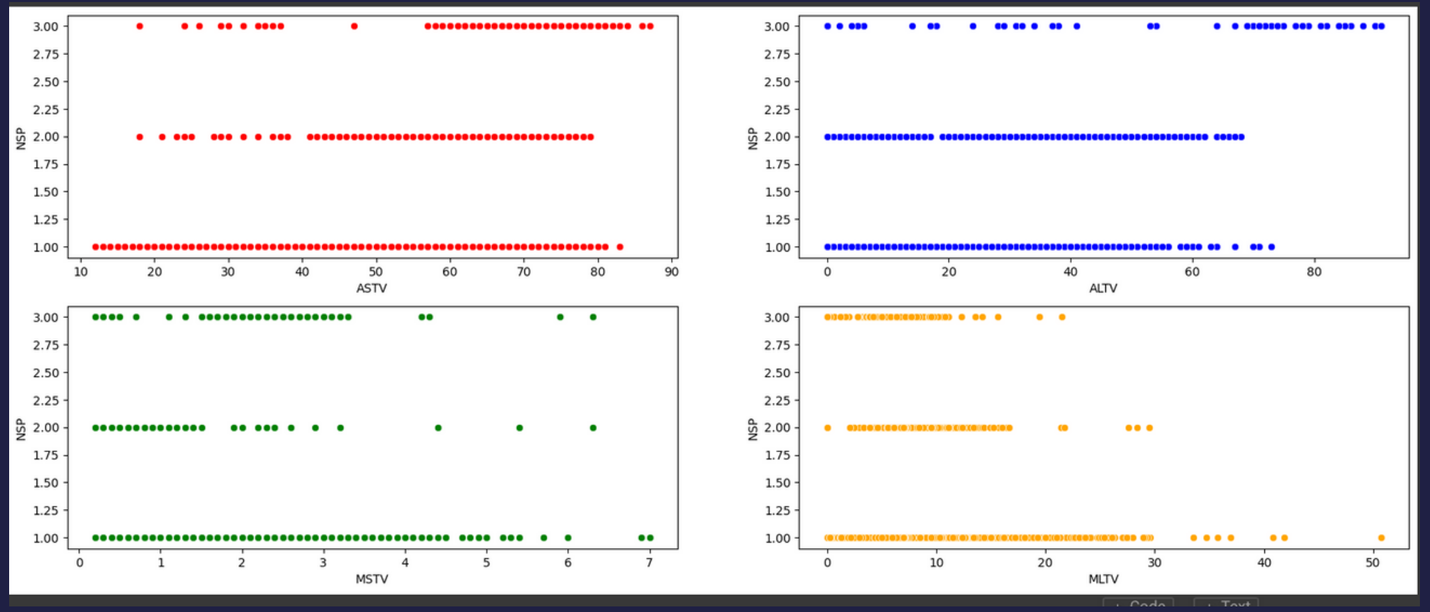
EDA

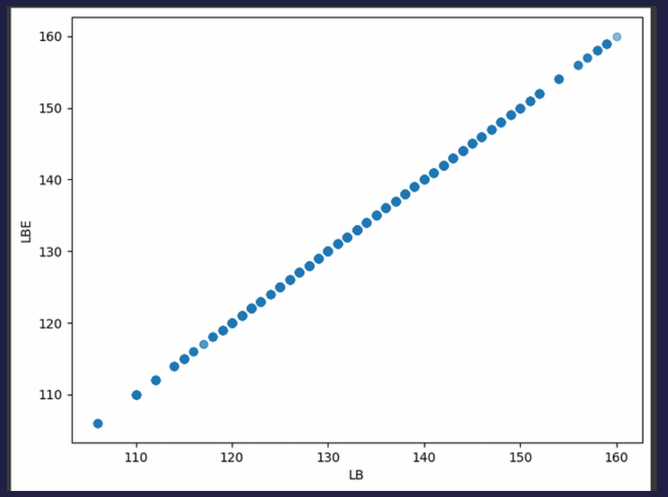
Heat Map

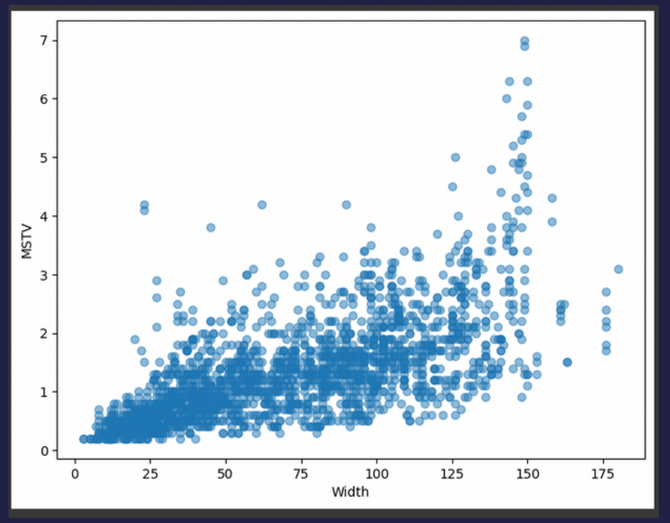

Heatplot for features
LBE LB AC FM UC ASTV MSTV ALTV MLTV DL DS DP Width Min Max Nmax Nzeros Mode Mean Median Variance Tendency CLASS NSP
1.00 0.75 0.50 0.25 0.00 −0.25 −0.50 −0.75

EDA

Scatter Plot

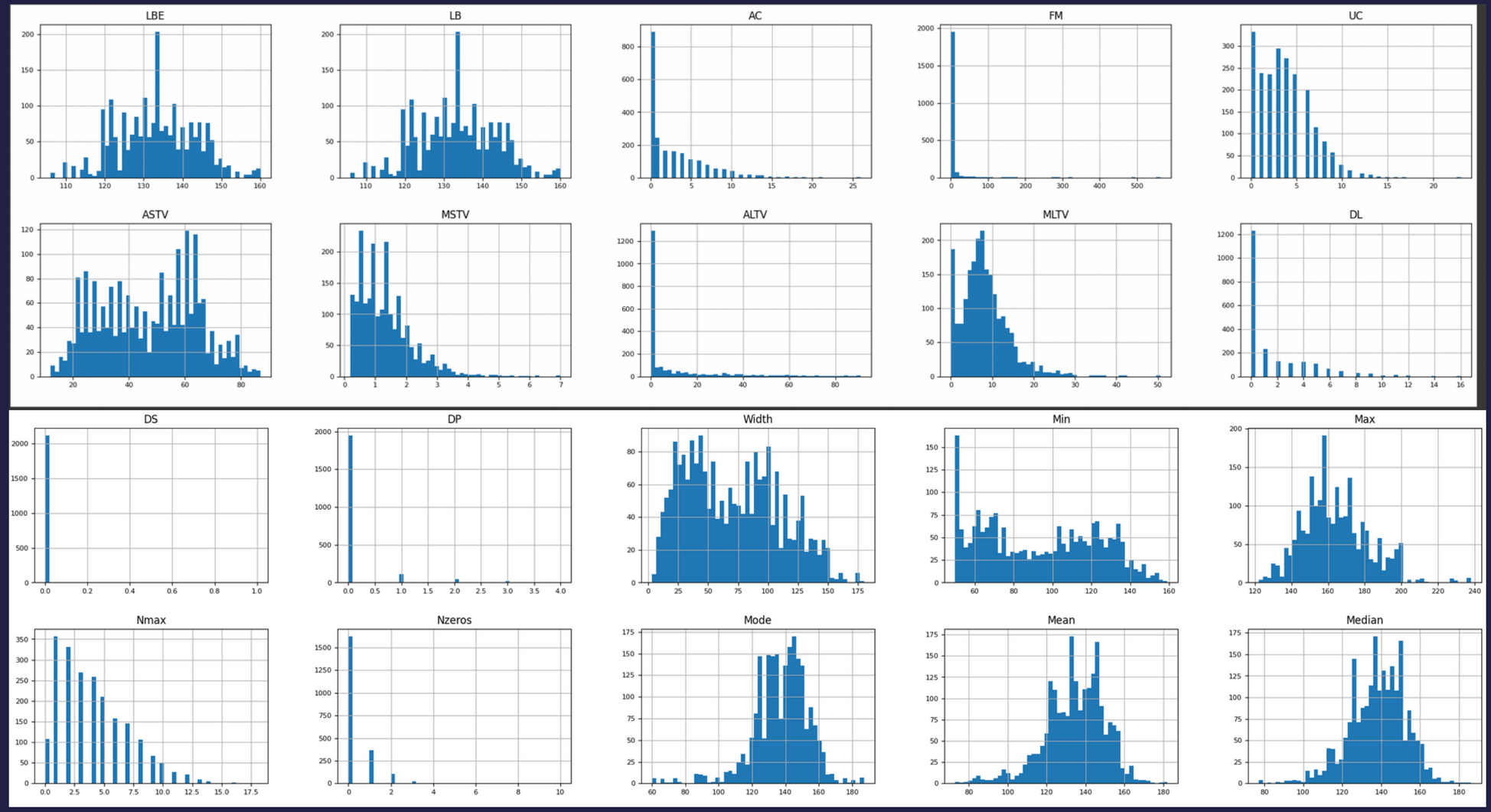






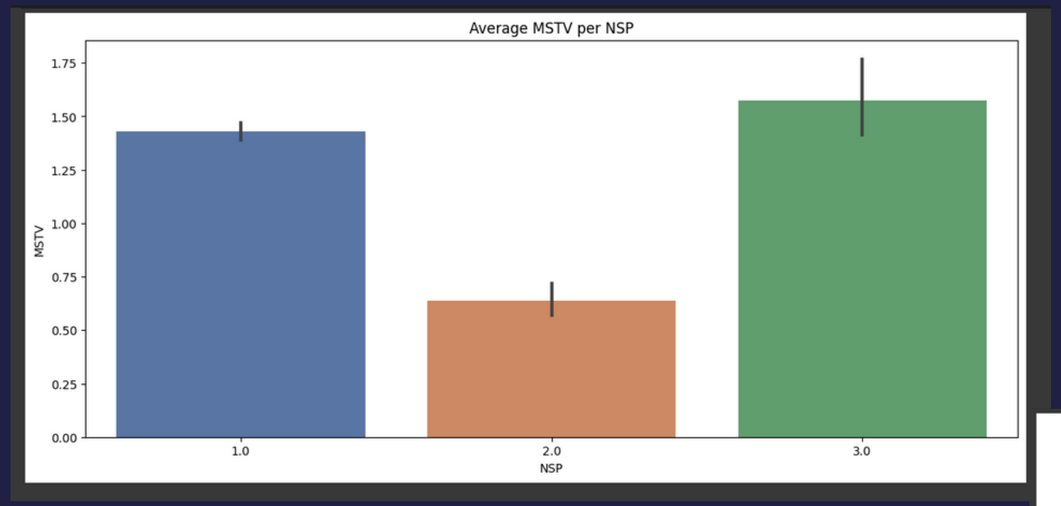
EDA

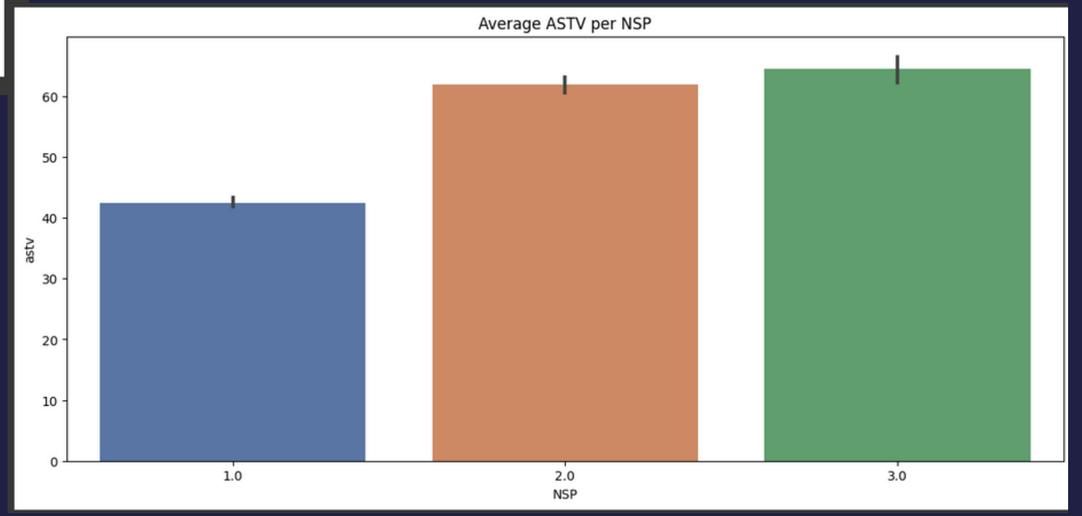
Distribusi Dataset dalam Barplot



EDA

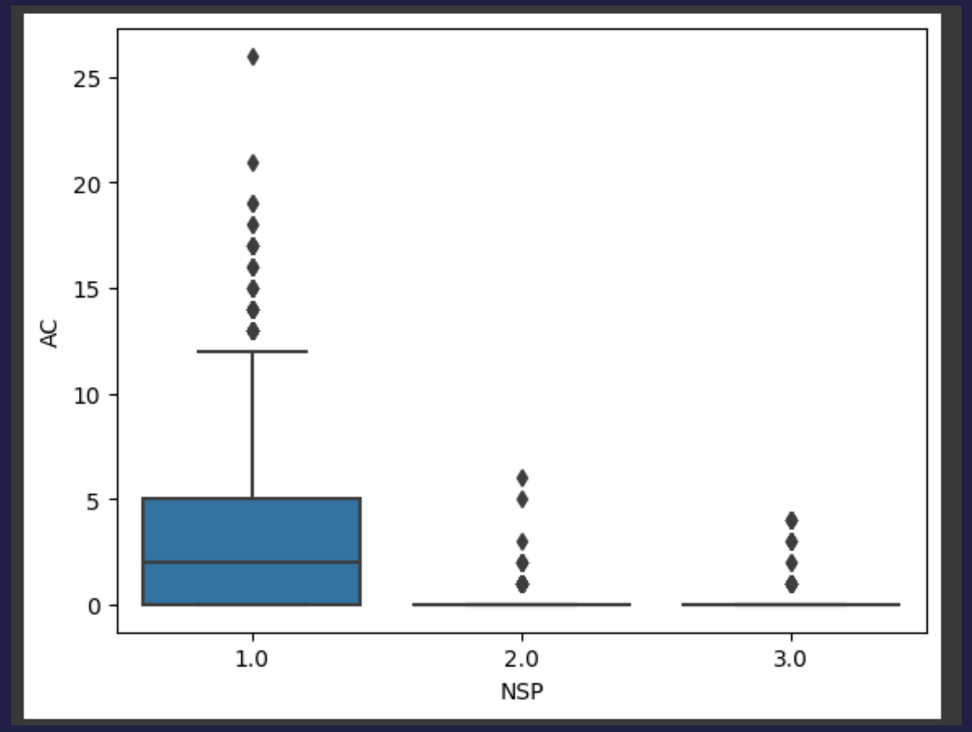
Rata-rata MSTV dan ASTV per NSP

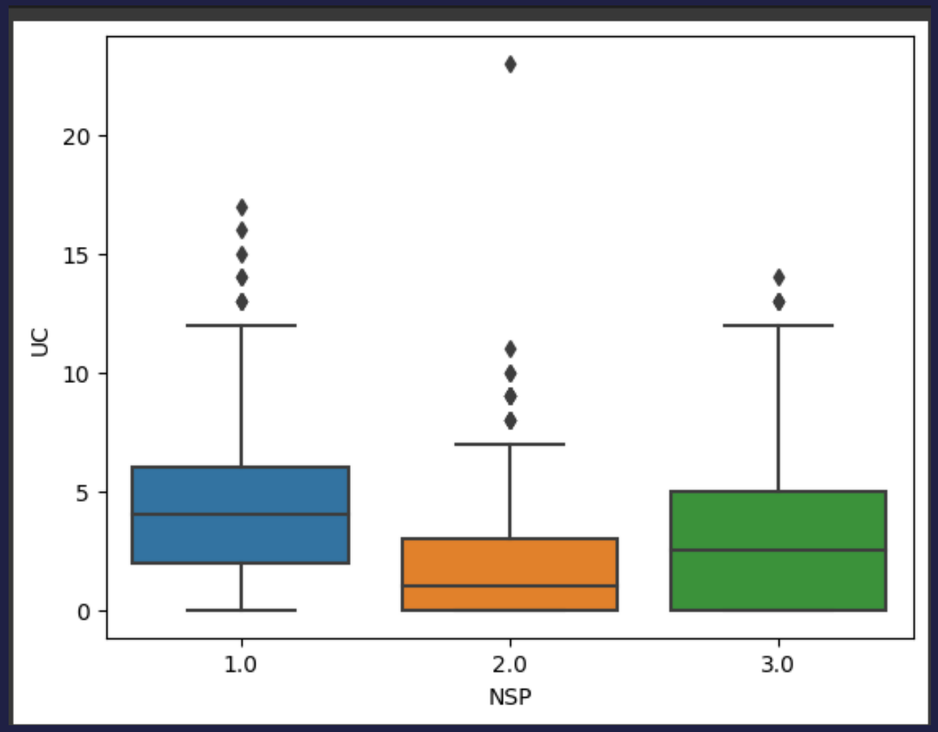




EDA

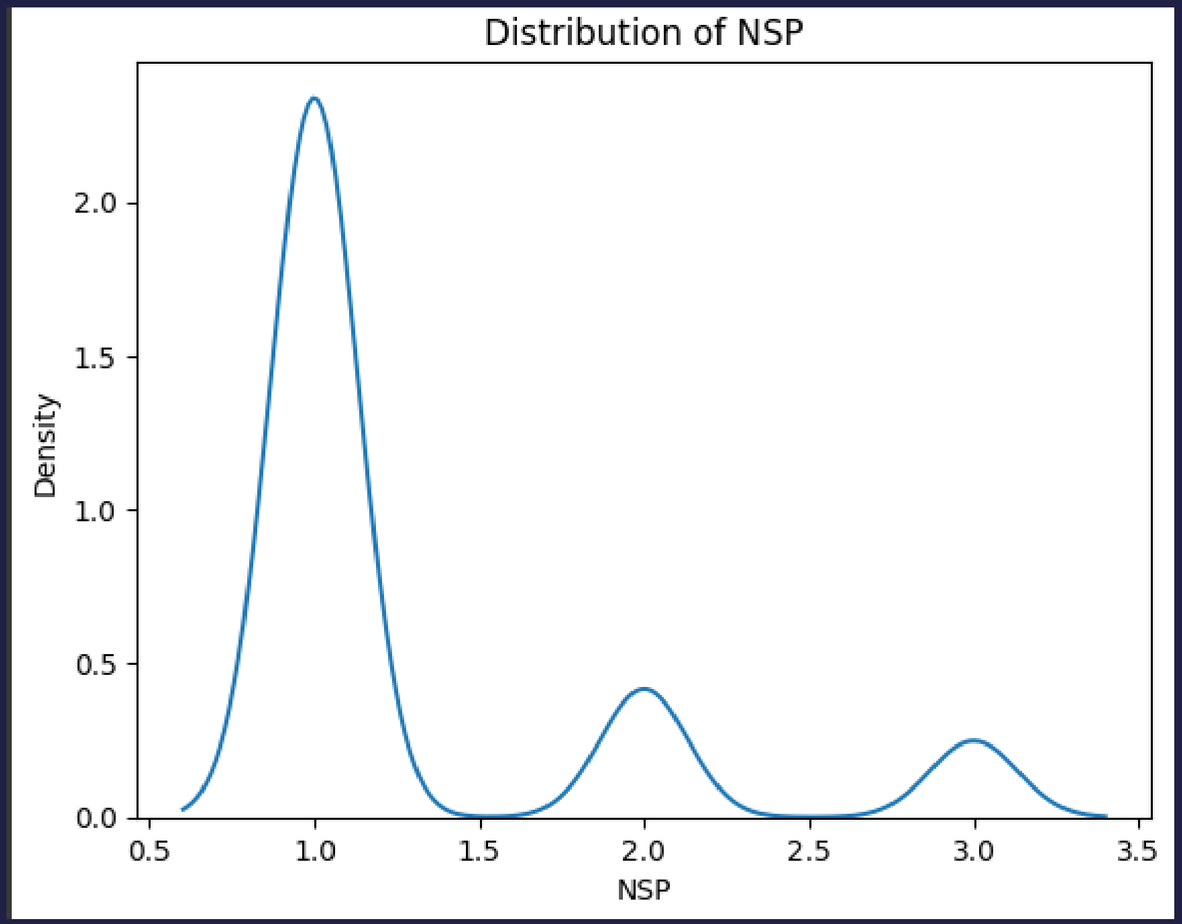
Distribusi AC dan UC per NSP





EDA

Distribusi NSP



Model Data

Model Decision Tree

Tahapan Proses Model Data

LANGKAH 4

Data Preprocessing

Decision Tree

Data Preprocessing

```
[ ]  # Dataset ini dapat digunakan untuk klasifikasi 10 kelas dan klasifikasi 3 kelas.
     # Memilih data yang akan digunakan untuk model 3 kelas
     X=df.drop(["NSP","CLASS"],axis=1)

     y=df["NSP"]
```

```
[ ]  X.head()
```

	LBE	LB	AC	FM	UC	ASTV	MSTV	ALTV	MLTV	DL	DS	DP	Width	Min	Max	Nmax	Nzeros	Mode	Mean	Median	Variance	Tendency
0	120.0	120.0	0.0	0.0	0.0	73.0	0.5	43.0	2.4	0.0	0.0	0.0	64.0	62.0	126.0	2.0	0.0	120.0	137.0	121.0	73.0	1.0
1	132.0	132.0	4.0	0.0	4.0	17.0	2.1	0.0	10.4	2.0	0.0	0.0	130.0	68.0	198.0	6.0	1.0	141.0	136.0	140.0	12.0	0.0
2	133.0	133.0	2.0	0.0	5.0	16.0	2.1	0.0	13.4	2.0	0.0	0.0	130.0	68.0	198.0	5.0	1.0	141.0	135.0	138.0	13.0	0.0
3	134.0	134.0	2.0	0.0	6.0	16.0	2.4	0.0	23.0	2.0	0.0	0.0	117.0	53.0	170.0	11.0	0.0	137.0	134.0	137.0	13.0	1.0
4	132.0	132.0	4.0	0.0	5.0	16.0	2.4	0.0	19.9	0.0	0.0	0.0	117.0	53.0	170.0	9.0	0.0	137.0	136.0	138.0	11.0	1.0

- Memilih data yang akan digunakan untuk model 3 kelas

- Melihat nilai unik dari 3 kelas yang ada

```
[ ]  nsp_classes = y.unique()
     nsp_classes
```

```
array([2., 1., 3.])
```

Data Preprocessing

```
[78] from keras import utils as np_utils
     from sklearn.preprocessing import LabelEncoder

     # Enkodekan nilai kelas sebagai bilangan bulat dan lakukan one-hot-encoding
     encoder = LabelEncoder()
     encoder.fit(y)
     z = encoder.transform(y)
     z = np_utils.to_categorical(y)

     z_new = z[:, 1:]
     print(z_new)
```

```
[[0. 1. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 ...
 [0. 1. 0.]
 [0. 1. 0.]
 [1. 0. 0.]]
```

- Menyimpan variabel baru untuk Enkodekan nilai kelas sebagai bilangan bulat dan lakukan one-hot-encoding

- Split Data Train dan Data Test pada Dataset

```
[82] from sklearn.model_selection import train_test_split

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

Data Preprocessing

```
[83] print(y_train.value_counts())

1.0    1323
2.0     236
3.0     141
Name: NSP, dtype: int64
```

Dibuat Data Buatan untuk menyeimbangkan NSP (balancing)

```
[84] from imblearn.under_sampling import RandomUnderSampler

rus = RandomUnderSampler(random_state=42)
X_train, y_train = rus.fit_resample(X_train, y_train)
```

```
[85] print(y_train.value_counts())

1.0    141
2.0    141
3.0    141
Name: NSP, dtype: int64
```

- Balancing Data untuk menyeimbangkan label NSP pada data train untuk mengurangi bias yang muncul

- X_train

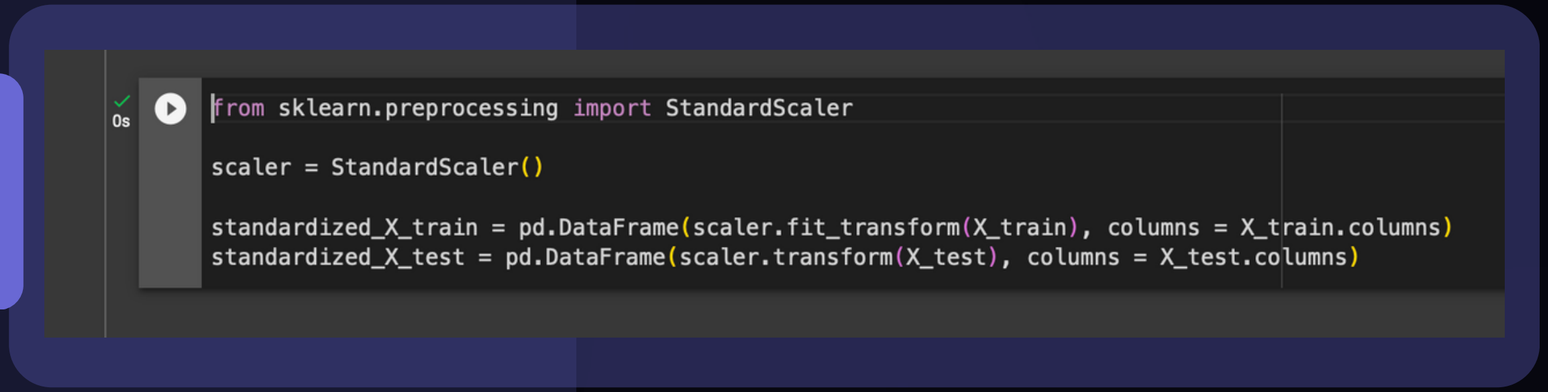
```
[86] X_train
```

	LBE	LB	AC	FM	UC	ASTV	MSTV	ALTV	MLTV	DL	DS	DP	Width	Min	Max	Nmax	Nzeros	Mode	Mean	Median	Variance	Tendency
0	140.0	140.0	0.0	0.0	6.0	79.0	0.3	20.0	8.5	0.0	0.0	0.0	26.0	124.0	150.0	1.0	0.0	144.0	143.0	145.0	1.0	1.0
1	134.0	134.0	0.0	0.0	2.0	22.0	2.1	0.0	8.0	3.0	0.0	0.0	111.0	54.0	165.0	5.0	2.0	126.0	118.0	127.0	59.0	0.0
2	133.0	133.0	1.0	0.0	2.0	33.0	1.4	3.0	12.8	9.0	0.0	0.0	87.0	65.0	152.0	6.0	0.0	136.0	130.0	137.0	26.0	1.0
3	128.0	128.0	3.0	10.0	7.0	24.0	2.0	0.0	9.0	5.0	0.0	1.0	91.0	65.0	156.0	6.0	0.0	133.0	125.0	131.0	42.0	1.0
4	132.0	132.0	6.0	0.0	6.0	27.0	1.7	0.0	11.3	0.0	0.0	0.0	122.0	54.0	176.0	8.0	0.0	150.0	146.0	149.0	18.0	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
418	134.0	134.0	3.0	0.0	0.0	64.0	1.9	0.0	0.0	1.0	0.0	3.0	120.0	66.0	186.0	9.0	0.0	88.0	104.0	106.0	109.0	-1.0
419	131.0	131.0	0.0	193.0	1.0	61.0	1.9	0.0	12.3	1.0	0.0	1.0	128.0	56.0	184.0	11.0	0.0	134.0	120.0	131.0	57.0	0.0
420	133.0	133.0	1.0	0.0	6.0	63.0	2.1	0.0	0.0	3.0	0.0	3.0	133.0	52.0	185.0	4.0	0.0	125.0	94.0	97.0	147.0	-1.0
421	128.0	128.0	0.0	0.0	0.0	80.0	0.5	0.0	6.8	0.0	0.0	0.0	16.0	114.0	130.0	0.0	0.0	126.0	124.0	125.0	1.0	1.0
422	128.0	128.0	0.0	219.0	2.0	34.0	2.5	0.0	4.0	2.0	0.0	2.0	145.0	54.0	199.0	11.0	1.0	75.0	99.0	102.0	148.0	-1.0

423 rows × 22 columns

Data Preprocessing

- StandardScaler untuk menyamakan skala fitur-fitur dalam data (normalisasi fitur)



Tahapan Proses Model Data

LANGKAH 5

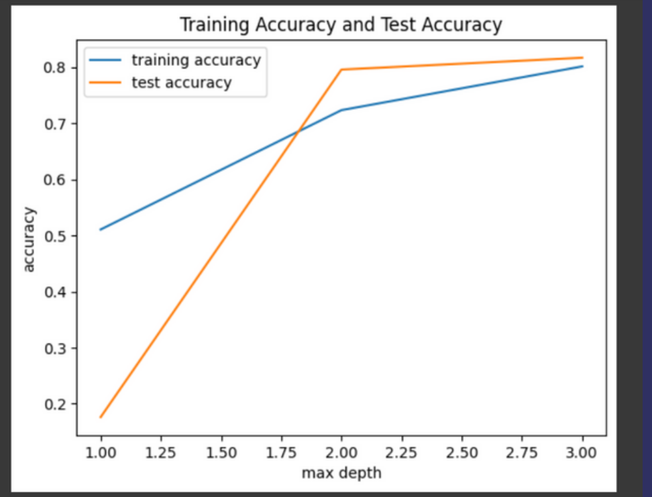
Model Selection

Decision Tree:
Kriteria Gini dan Entropi

Model Selection

Kriteria Gini

Training Accuracy dan Test Accuracy



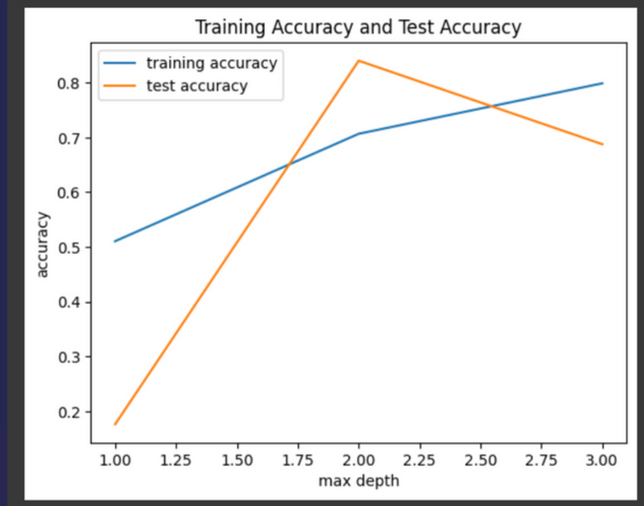
F1-Score dan Overall Accuracy

```
[91] from sklearn.metrics import f1_score

     f1 = f1_score(y_test, y_pred_test, average='weighted')
     print("F1-score:", f1)

     overall_accuracy = accuracy_score(y_test, y_pred_test)
     print("Overall Accuracy:", overall_accuracy)
```

```
F1-score: 0.8320205430212965
Overall Accuracy: 0.8169014084507042
```

Kriteria Entropi

Training Accuracy dan Test Accuracy



F1-Score dan Overall Accuracy

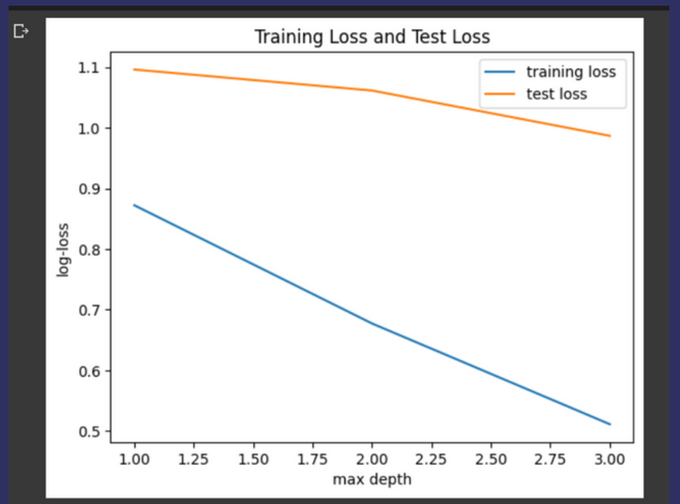
```
[105] from sklearn.metrics import f1_score

      f1 = f1_score(y_test, y_pred_test, average='weighted')
      print("F1-score:", f1)

      overall_accuracy = accuracy_score(y_test, y_pred_test)
      print("Overall Accuracy:", overall_accuracy)
```

```
F1-score: 0.7330293498882337
Overall Accuracy: 0.687793427230047
```

Model Selection

Kriteria Gini

Plotting Training Loss dan Test Loss

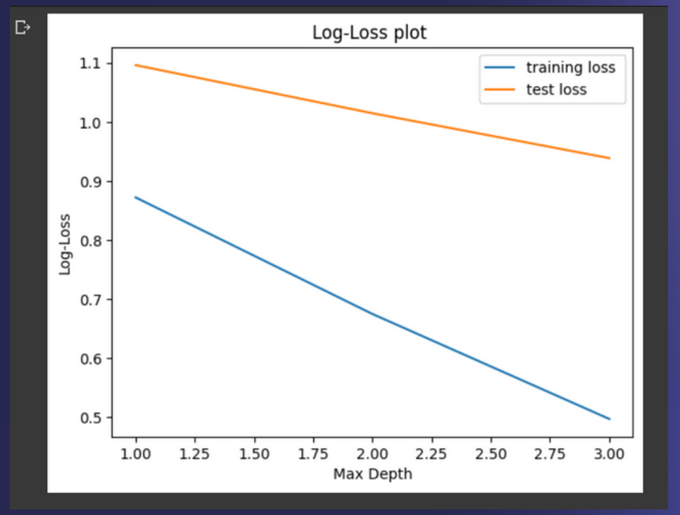


Code Training loss dan Test Loss

```
from sklearn.metrics import log_loss
from sklearn.tree import DecisionTreeClassifier


training_loss = []
test_loss = []

def tree_scores(i):
    clf_gini = DecisionTreeClassifier(criterion='gini', max_depth=i, random_state = 42)


    clf_gini.fit(standardized_X_train, y_train)
    y_pred_gini = clf_gini.predict_proba(standardized_X_test)
    y_pred_train_gini = clf_gini.predict_proba(standardized_X_train)


    training_loss.append(log_loss(y_train, y_pred_train_gini))

    test_loss.append(log_loss(y_test, y_pred_gini))


for i in range(1,4):
    tree_scores(i)
```

Kriteria Entropi

Plotting Training Loss dan Test Loss



Code Training Loss dan Test Loss

```
[106] training_loss = []
      test_loss = []

      def tree_scores(i):
          clf_en = DecisionTreeClassifier(criterion='entropy', max_depth=i, random_state = 42)


          clf_en.fit(standardized_X_train, y_train)
          y_pred_en = clf_en.predict_proba(standardized_X_test)
          y_pred_train_en = clf_en.predict_proba(standardized_X_train)


          training_loss.append(log_loss(y_train, y_pred_train_en))

          test_loss.append(log_loss(y_test, y_pred_en))


      for i in range(1,4):
          tree_scores(i)
```

Tahapan Proses Model Data

LANGKAH 6

Hyperparameter Tuning

Decision Tree:
Kriteria Gini dan Entropi

Hyperparameter Tuning

Kriteria Gini

Cross-Validation Score

Hyperparametertuning

```
[94] from sklearn.model_selection import cross_val_score
     clf_gini = DecisionTreeClassifier(criterion='gini', max_depth=6, random_state = 42)
     print('Cross-Validation Score:',np.mean(cross_val_score(clf_gini, standardized_X_train, y_train, cv=20)))
     Cross-Validation Score: 0.8774891774891775
```

Kriteria Entropi

Cross-Validation Score

Hyperparameter Tuning

```
[108] clf_en = DecisionTreeClassifier(criterion='entropy', max_depth=4, random_state = 42)
      print('Cross-Validation Score:',np.mean(cross_val_score(clf_en, standardized_X_train, y_train, cv=20)))
      Cross-Validation Score: 0.8561688311688312
```

Tahapan Proses Model Data

LANGKAH 7

Model Evaluation

Decision Tree:
Kriteria Gini dan Entropi

Model Evaluation

Kriteria Gini

F1-Score dan Accuracy

```
clf_gini = DecisionTreeClassifier(criterion='gini', max_depth=6, random_state = 42)

clf_gini.fit(standardized_X_train, y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=6, random_state=42)
```

```
[96] y_pred_gini = clf_gini.predict(standardized_X_test)
```

```
[97] from sklearn.metrics import accuracy_score
     from sklearn.metrics import f1_score

     f1 = f1_score(y_test, y_pred_gini, average='weighted')
     print("F1-score:", f1)

     print("Accuracy:",accuracy_score(y_test, y_pred_gini))
```

```
F1-score: 0.8732138491163518
Accuracy: 0.8685446009389671
```

Kriteria Entropi

F1-Score dan Accuracy

```
clf_en = DecisionTreeClassifier(criterion='entropy', max_depth=4, random_state = 42)

clf_en.fit(standardized_X_train, y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=4, random_state=42)
```

```
[110] y_pred_en = clf_en.predict(standardized_X_test)
```

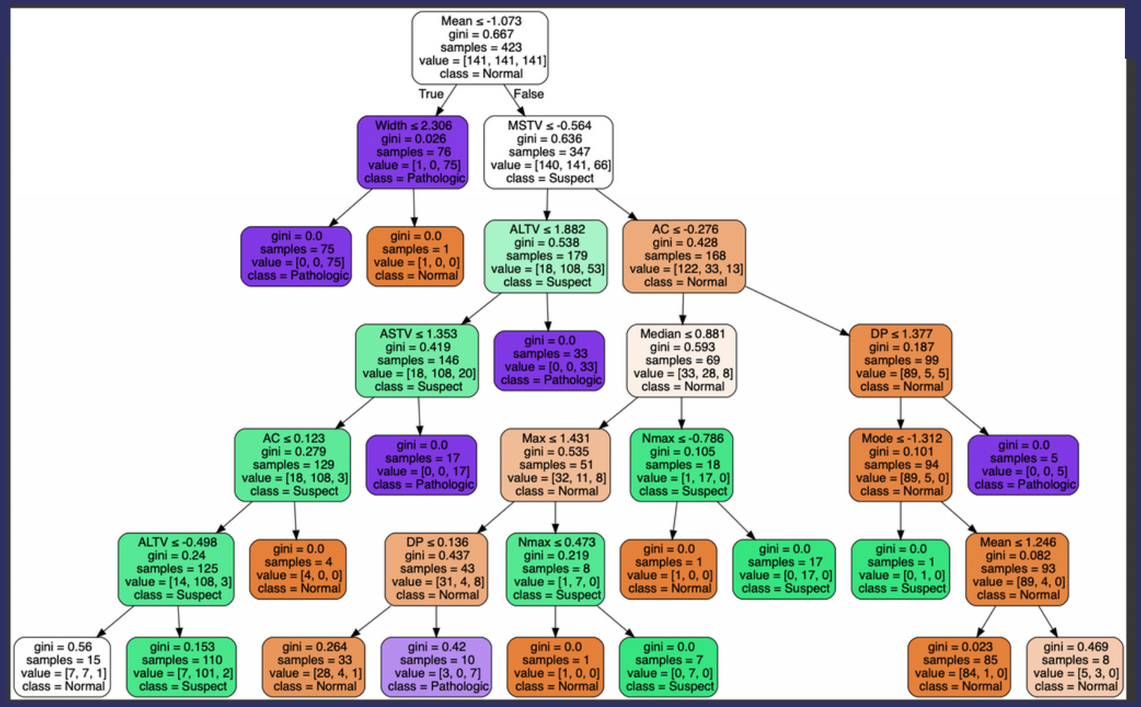
```
[111] f1 = f1_score(y_test, y_pred_en, average='weighted')
      print("F1-score:", f1)
      print("Accuracy:",accuracy_score(y_test, y_pred_en))
```

```
F1-score: 0.8352388849680289
Accuracy: 0.8169014084507042
```

Model Evaluation

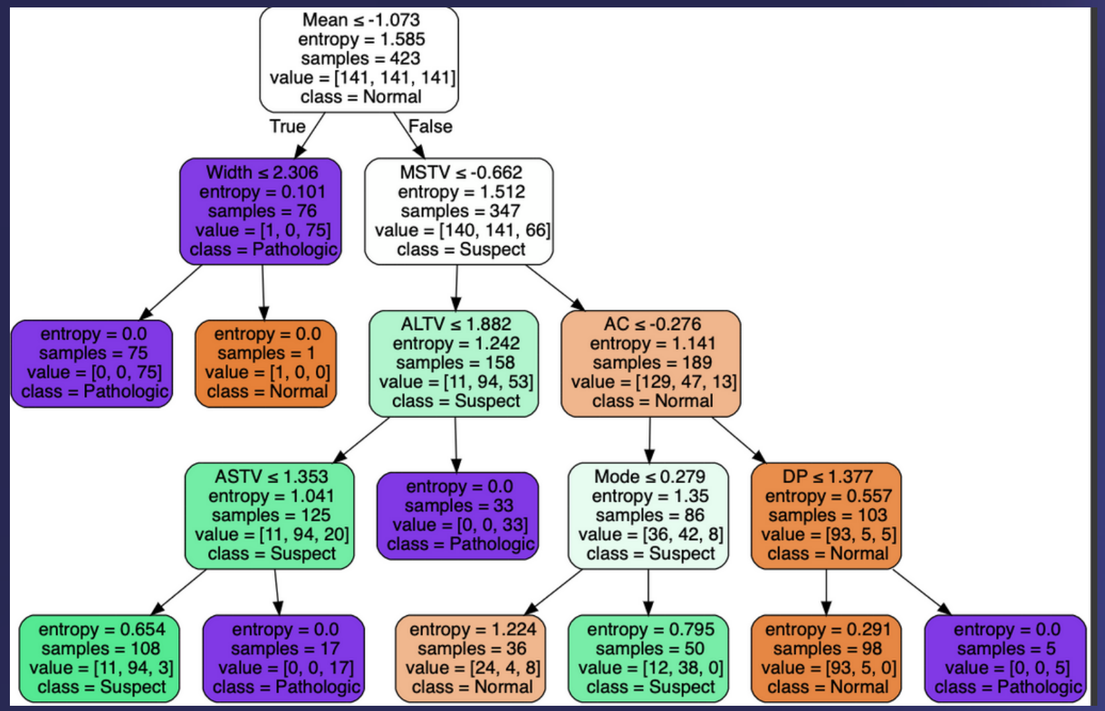
Kriteria Gini

Decision Tree



Kriteria Entropi

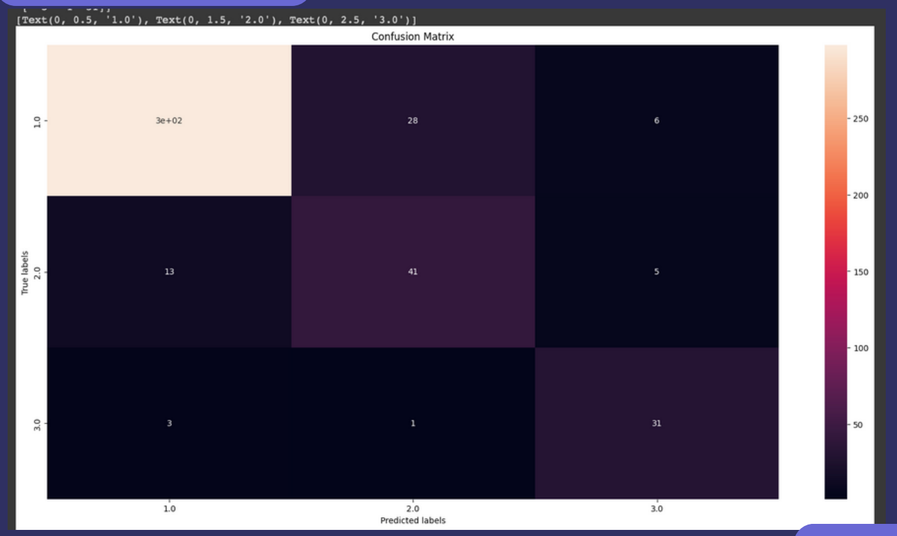
Decision Tree



Model Evaluation

Kriteria Gini

Confusion Matrix



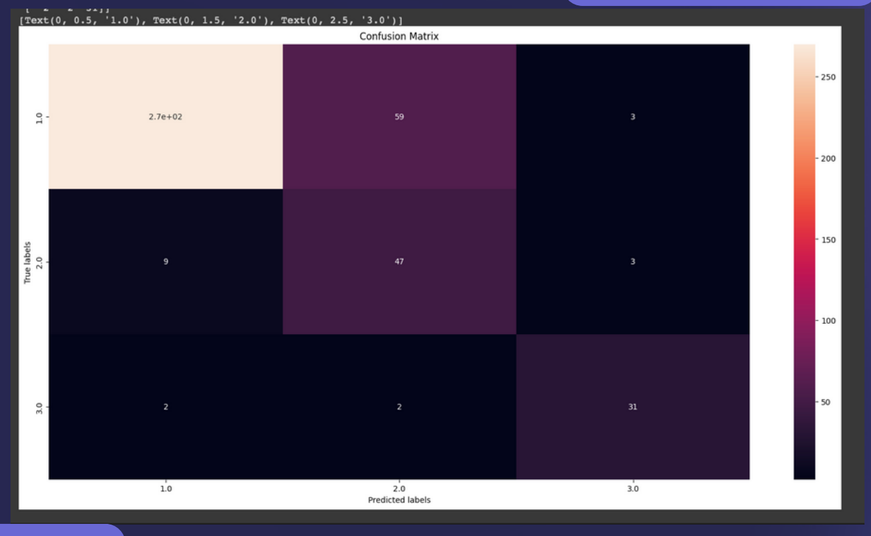
Classification Report

```
✓[102]from sklearn.metrics import classification_report
0s    print(classification_report(y_test, y_pred_gini))
```

	precision	recall	f1-score	support
1.0	0.95	0.90	0.92	332
2.0	0.59	0.69	0.64	59
3.0	0.74	0.89	0.81	35
accuracy			0.87	426
macro avg	0.76	0.83	0.79	426
weighted avg	0.88	0.87	0.87	426

Kriteria Entropi

Confusion Matrix



Classification Report

```
✓[114]print(classification_report(y_test, y_pred_en))
0s
```

	precision	recall	f1-score	support
1.0	0.96	0.81	0.88	332
2.0	0.44	0.80	0.56	59
3.0	0.84	0.89	0.86	35
accuracy			0.82	426
macro avg	0.74	0.83	0.77	426
weighted avg	0.88	0.82	0.84	426

Model Data

Model Keras ANN

Tahapan Proses Model Data

LANGKAH 4

Data Preprocessing

Keras

Data Preprocessing

```
[ ]  # Dataset ini dapat digunakan untuk klasifikasi 10 kelas dan klasifikasi 3 kelas.
     # Memilih data yang akan digunakan untuk model 3 kelas
     X=df.drop(["NSP","CLASS"],axis=1)

     y=df["NSP"]
```

```
[ ]  X.head()
```

	LBE	LB	AC	FM	UC	ASTV	MSTV	ALTV	MLTV	DL	DS	DP	Width	Min	Max	Nmax	Nzeros	Mode	Mean	Median	Variance	Tendency
0	120.0	120.0	0.0	0.0	0.0	73.0	0.5	43.0	2.4	0.0	0.0	0.0	64.0	62.0	126.0	2.0	0.0	120.0	137.0	121.0	73.0	1.0
1	132.0	132.0	4.0	0.0	4.0	17.0	2.1	0.0	10.4	2.0	0.0	0.0	130.0	68.0	198.0	6.0	1.0	141.0	136.0	140.0	12.0	0.0
2	133.0	133.0	2.0	0.0	5.0	16.0	2.1	0.0	13.4	2.0	0.0	0.0	130.0	68.0	198.0	5.0	1.0	141.0	135.0	138.0	13.0	0.0
3	134.0	134.0	2.0	0.0	6.0	16.0	2.4	0.0	23.0	2.0	0.0	0.0	117.0	53.0	170.0	11.0	0.0	137.0	134.0	137.0	13.0	1.0
4	132.0	132.0	4.0	0.0	5.0	16.0	2.4	0.0	19.9	0.0	0.0	0.0	117.0	53.0	170.0	9.0	0.0	137.0	136.0	138.0	11.0	1.0

- Memilih data yang akan digunakan untuk model 3 kelas

- Melihat nilai unik dari 3 kelas yang ada

```
[ ]  nsp_classes = y.unique()
     nsp_classes
```

```
array([2., 1., 3.])
```

Data Preprocessing

```
[ ]  from keras import utils as np_utils
     from sklearn.preprocessing import LabelEncoder

     # Enkodekan nilai kelas sebagai bilangan bulat dan lakukan one-hot-encoding
     encoder = LabelEncoder()
     encoder.fit(y)
     y = encoder.transform(y)
     y = np_utils.to_categorical(y)
     print(y)

     y.shape
```

```
[[0. 1. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 ...
 [0. 1. 0.]
 [0. 1. 0.]
 [1. 0. 0.]]
(2126, 3)
```

- Melakukan metode One-Hot-Encoding

- Melakukan standarisasi data dengan Standard Scaler

```
[ ]  # Standardisasi Data dengan Standard Scaler
     from sklearn.preprocessing import StandardScaler
     Scaler=StandardScaler()
     X=Scaler.fit_transform(X)

     X[0:3]

     X.shape
```

```
(2126, 22)
```

Tahapan Proses Model Data

LANGKAH 5

Model Selection

Keras

Model Selection

```
#Import module yang akan dipakai pada pembuatan model
from sklearn.model_selection import GridSearchCV
from keras.wrappers.scikit_learn import KerasClassifier
from keras.models import Sequential
from keras.layers import Dense


#Split data menjadi data train dan test
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                                    random_state=0)
```

```
#Pemberian bobot pada tiap kelas supaya mengurangi ketidakseimbangan
# saat train data
class_weight = {0: 1, 1: 5.74, 2: 9.4}
```

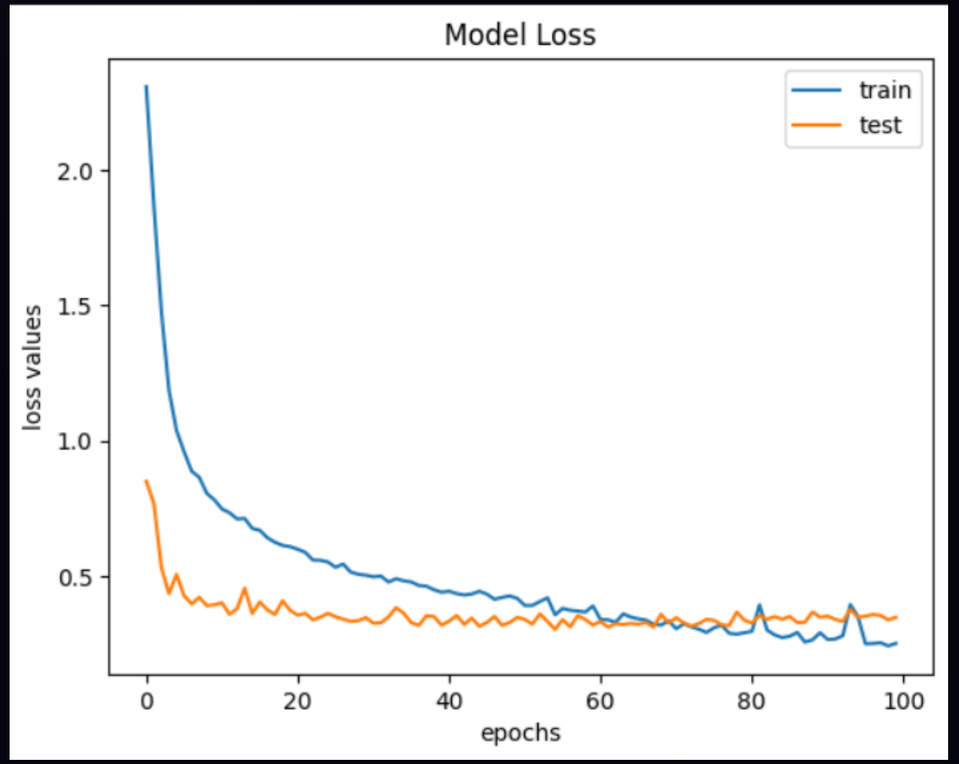
Model Selection

```
#Mendefinisikan pembuatan model dengan keras sequential
def create_model(optimizer="adam"):
    #membuat model
    model = Sequential()
    model.add(Dense(20, input_dim=22, activation='relu'))
    model.add(Dense(40, activation='sigmoid'))
    model.add(Dense(60, activation='relu'))
    model.add(Dense(3, activation='softmax'))
    #Layer terakhir berisikan 3 karena outputnya 3

    #Karena multi-class, maka fungsi aktivasi "softmax" yang dipilih
    #dan fungsi loss nya "categorical_crossentropy"

    #Compile model
    model.compile(loss='categorical_crossentropy',
                  optimizer=optimizer,metrics=["accuracy"])
    return model

model = create_model()
```

Model Selection

```
#Memfit train yang akan digunakan pada data
train = model.fit(X_train, y_train, epochs=100, batch_size=32,
                  class_weight=class_weight, verbose=1,
                  validation_data=(X_test,y_test))
```

```
#Mengeplot model loss ketika train data
#Untuk mengetahui apakah data train sudah bagus atau belum
import matplotlib.pyplot as plt
plt.plot(train.history['loss'], label='train')
plt.plot(train.history['val_loss'], label='test')
plt.title('Model Loss')
plt.xlabel('epochs')
plt.ylabel('loss values')
plt.legend(loc='upper right')
plt.show()
```



Model Selection

```
#Memodelkan y prediksi
import sklearn.metrics as metrics
y_pred = model.predict(X_test)
```

```
#Mengecek f1 score dan akurasi pada model
#f1 score
import sklearn.metrics as metrics
print("f1 scores:",metrics.f1_score(np.argmax(y_test, axis=1),
                                    np.argmax(y_pred, axis=1),
                                    average='weighted'))

#Accuracy
print("accuracy:",metrics.accuracy_score(np.argmax(y_test, axis=1),
                                         np.argmax(y_pred, axis=1)))
```

```
f1 scores: 0.9072238610038444
accuracy: 0.9014084507042254
```

Tahapan Proses Model Data

LANGKAH 6

Hyperparameter Tuning

Keras

Hyperparameter Tuning

```
#Mentuning model dengan Grid Search Cross Validation
#Parameter GridSearch Cross Validation
param_grid = {
    'epochs': [50,100,150],
    'batch_size':[32,50,100],
    'optimizer':['RMSprop', 'Adam','SGD'],
}

#Membuat model yang akan dituning
#Membuat objek model dengan KerasClassifier
model_cv = KerasClassifier(build_fn=create_model, verbose=1)

grid = GridSearchCV(estimator=model_cv,
                    n_jobs=-1,
                    verbose=1,
                    cv=5,
                    param_grid=param_grid)
grid_cv_model = grid.fit(X_train, y_train,) #Memfit object grid search pada data tr


means = grid_cv_model.cv_results_['mean_test_score'] #Mean dari skor test
stds = grid_cv_model.cv_results_['std_test_score'] #Standar devisiasi dari skor tes
params = grid_cv_model.cv_results_['params'] #Menggunakan parameter

#Mengeprint semua skor, standar devisiasi dan parameter
#yang digunakan untuk mencari parameter terbaik
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))

#Mengeprint Parameter terbaik yang digunakan untuk hasil Grid Search Cross Validati
print("Best: %f using %s" % (grid_cv_model.best_score_, grid_cv_model.best_params_)
```

```
#Membuat model yang akan dituning dengan parameter terbaik yang telah diketahui
 #Pembuatan objek model yang telah di tuning menggunakan KerasClassifier
cv_model = grid_cv_model.best_estimator_
```

Melakukan tuning model dengan Grid Search Cross Validation dan mencari parameter terbaik dari beberapa parameter yang ditentukan.

Diperoleh parameter terbaik dengan

- Batch Size : 32
- Epochs : 150
- Optimizer Adam

K-fold Cross Validation

```
#K-FOLD
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score

#K-fold accuracy scores
kfold = KFold(n_splits=5, shuffle=True)
results = cross_val_score(cv_model, X_test, np.argmax(y_test, axis=1),
                          cv=kfold,scoring= 'accuracy')

print('K-fold Cross Validation Accuracy Results: ', results)
print('K-fold Cross Validation Accuracy Results Mean: ', results.mean())
```

```
K-fold Cross Validation Accuracy Results:  [0.90697674 0.88235294 0.85882353 0.85882
K-fold Cross Validation Accuracy Results Mean:  0.8708071135430917
```

```
#K-fold f1 scores
from sklearn.model_selection import KFold


kfold = KFold(n_splits=5, shuffle=True)
results = cross_val_score(cv_model, X_test, np.argmax(y_test, axis=1),
                          cv=kfold,scoring="f1_weighted")

print('K-fold Cross Validation f1_weighted Results: ', results)
print('K-fold Cross Validation f1_weighted Results Mean: ', results.mean())
```

```
K-fold Cross Validation f1_weighted Results:  [0.8684547  0.86178345 0.86440299 0.86
K-fold Cross Validation f1_weighted Results Mean:  0.8669106760060588
```

Tahapan Proses Model Data

LANGKAH 7

Model Evaluation

Keras

Model Evaluation

- Nilai final F1 Score dan Akurasi

```
import sklearn.metrics as metrics
#f1 score
print("f1_weighted:",metrics.f1_score(np.argmax(y_test, axis=1),
                                      y_pred,average='weighted'))

#Accuracy
print("accuracy:",metrics.accuracy_score(np.argmax(y_test, axis=1), y_pred))
```

```
f1_weighted: 0.9144984296615148
accuracy: 0.9131455399061033
```

- Confusion Matrix

```
# Confusion Matrix
model_conf = confusion_matrix(np.argmax(y_test, axis=1), y_pred)
print(model_conf)
```

```
[[309  15   2]
 [ 13  43   2]
 [  0   5  37]]
```

Model Evaluation

```python
#Confusion Matrix dan Classification Report
from sklearn.metrics import confusion_matrix, classification_report

#Classification Report
model_report = classification_report(np.argmax(y_test, axis=1), y_pred)
print(model_report)
```

```
              precision    recall  f1-score   support

           0       0.96      0.95      0.95       326
           1       0.68      0.74      0.71        58
           2       0.90      0.88      0.89        42

    accuracy                           0.91       426
   macro avg       0.85      0.86      0.85       426
weighted avg       0.92      0.91      0.91       426
```

- Classification Report

Model Data

Model
Multilayer Perceptron

Tahapan Proses Model Data

LANGKAH 4

Data Preprocessing

Multilayer Perceptron

Data Preprocessing

```
[ ]  # Dataset ini dapat digunakan untuk klasifikasi 10 kelas dan klasifikasi 3 kelas.
     # Memilih data yang akan digunakan untuk model 3 kelas
     X=df.drop(["NSP","CLASS"],axis=1)

     y=df["NSP"]
```

```
[ ]  X.head()
```

	LBE	LB	AC	FM	UC	ASTV	MSTV	ALTV	MLTV	DL	DS	DP	Width	Min	Max	Nmax	Nzeros	Mode	Mean	Median	Variance	Tendency
0	120.0	120.0	0.0	0.0	0.0	73.0	0.5	43.0	2.4	0.0	0.0	0.0	64.0	62.0	126.0	2.0	0.0	120.0	137.0	121.0	73.0	1.0
1	132.0	132.0	4.0	0.0	4.0	17.0	2.1	0.0	10.4	2.0	0.0	0.0	130.0	68.0	198.0	6.0	1.0	141.0	136.0	140.0	12.0	0.0
2	133.0	133.0	2.0	0.0	5.0	16.0	2.1	0.0	13.4	2.0	0.0	0.0	130.0	68.0	198.0	5.0	1.0	141.0	135.0	138.0	13.0	0.0
3	134.0	134.0	2.0	0.0	6.0	16.0	2.4	0.0	23.0	2.0	0.0	0.0	117.0	53.0	170.0	11.0	0.0	137.0	134.0	137.0	13.0	1.0
4	132.0	132.0	4.0	0.0	5.0	16.0	2.4	0.0	19.9	0.0	0.0	0.0	117.0	53.0	170.0	9.0	0.0	137.0	136.0	138.0	11.0	1.0

- Memilih data yang akan digunakan untuk model 3 kelas

- Melihat nilai unik dari 3 kelas yang ada

```
[ ]  nsp_classes = y.unique()
     nsp_classes
```

```
array([2., 1., 3.])
```

Data Preprocessing

```
[ ]  from keras import utils as np_utils
     from sklearn.preprocessing import LabelEncoder

     # Enkodekan nilai kelas sebagai bilangan bulat dan lakukan one-hot-encoding
     encoder = LabelEncoder()
     encoder.fit(y)
     y = encoder.transform(y)
     y = np_utils.to_categorical(y)
     print(y)

     y.shape
```

```
[[0. 1. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 ...
 [0. 1. 0.]
 [0. 1. 0.]
 [1. 0. 0.]]
(2126, 3)
```

- Melakukan metode One-Hot-Encoding

- Melakukan standarisasi data dengan Standard Scaler

```
[ ]  # Standardisasi Data dengan Standard Scaler
     from sklearn.preprocessing import StandardScaler
     Scaler=StandardScaler()
     X=Scaler.fit_transform(X)

     X[0:3]

     X.shape
```

```
(2126, 22)
```

Tahapan Proses Model Data

LANGKAH 5

Model Selection

Multilayer Perceptron

Model Selection

```
[ ]  from sklearn.model_selection import train_test_split
     # Melakukan split data menjadi train dan test
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                                         random_state=0)
```

```
[ ]  # Multi Layer Perceptron Artificial Neural Network
     from sklearn.neural_network import MLPClassifier

     mlpc = MLPClassifier(random_state = 0) # Membuat objek model ANN

     mlpc.fit(X_train, y_train) # ANN model object fit
```

```
▾ MLPClassifier
MLPClassifier(random_state=0)
```

Model Selection

```
[ ]  # Peramalan pada Model yang Tidak Divalidasi
     y_pred = mlpc.predict(X_test) # Proses prediksi model atas set tes
```

```
[ ]  import sklearn.metrics as metrics

     # Akurasi

     print("Accuracy:",metrics.accuracy_score(y_test,y_pred))

     # f1 score

     print("f1_weighted:",metrics.f1_score(y_test, y_pred,average='weighted'))
```

```
Accuracy: 0.8967136150234741
f1_weighted: 0.9146198501918394
```

Tahapan Proses Model Data

LANGKAH 6

Hyperparameter Tuning

Multilayer Perceptron

Hyperparameter Tuning

```
[ ]  # Cross Validation Process
     mlpc_params = {"alpha": [0.1, 0.01, 0.0001],
                    "hidden_layer_sizes": [(10,10,10),
                                           (100,100,100),
                                           (100,100)],
                    "solver" : ["lbfgs","adam","sgd"],
                    "activation": ["relu","logistic"]}

     from sklearn.model_selection import GridSearchCV

     mlpc = MLPClassifier(random_state = 0) # Membuat objek model ANN

     # Model CV process
     mlpc_cv_model = GridSearchCV(mlpc, mlpc_params,
                                  cv = 5, # Untuk membuat 5-fold CV
                                  n_jobs = -1, # Jumlah pekerjaan yang akan dijalankan secara paralel
                                  verbose = 2) # Level kontrol dari detail

     mlpc_cv_model.fit(X_train, y_train)


     # Parameter terbaik diperoleh dari hasil proses CV

     print("The best parameters: " + str(mlpc_cv_model.best_params_))
```

```
Fitting 5 folds for each of 54 candidates, totalling 270 fits
The best parameters: {'activation': 'relu', 'alpha': 0.01, 'hidden_layer_sizes': (100, 100, 100), 'solver': 'adam'}
```

Grid Search
Cross Validation

Hyperparameter Tuning

```
[ ]  # Model Tuning
     # Setting Final Model dengan parameter terbaik

     mlpc_tuned = mlpc_cv_model.best_estimator_

     # Pemasangan model final
     mlpc_tuned.fit(X_train, y_train)
```

```
MLPClassifier
MLPClassifier(alpha=0.01, hidden_layer_sizes=(100, 100, 100), random_state=0)
```

Grid Search
Cross Validation

Hyperparameter Tuning

K-Fold
Cross Validation

```
[ ]  # K-fold accuracy

     from sklearn.model_selection import KFold
     from sklearn.model_selection import cross_val_score

     # K fold
     kf = KFold(shuffle=True, n_splits=5) # Untuk membuat 5-fold CV

     cv_results_kfold = cross_val_score(mlpc_tuned, X_test, np.argmax(y_test, axis=1), cv=kf, scoring= 'accuracy')

     print("K-fold Cross Validation accuracy Results: ",cv_results_kfold)
     print("K-fold Cross Validation accuracy Results Mean: ",cv_results_kfold.mean())
```

```
K-fold Cross Validation accuracy Results:  [0.80232558 0.85882353 0.84705882 0.88235294 0.90588235]
K-fold Cross Validation accuracy Results Mean:  0.8592886456908344
```

```
[ ]  # Tune Model Prediction
     # Proses prediksi model final terhadap test
     y_pred = mlpc_tuned.predict(X_test)
```

Tahapan Proses Model Data

LANGKAH 7

Model Evaluation

Multilayer Perceptron

Model Evaluation

Nilai final
F1 Score dan Akurasi

```
[ ]  # Nilai akurasi adan f1_weighted untuk model final

     # %% f1 score
     import sklearn.metrics as metrics
     print("f1_weighted:",metrics.f1_score(np.argmax(y_test, axis=1), np.argmax(y_pred, axis=1),average='weighted'))

     # %% Akurasi

     print("accuracy:",metrics.accuracy_score(np.argmax(y_test, axis=1), np.argmax(y_pred, axis=1)))
```

```
f1_weighted: 0.9170465379635293
accuracy: 0.9178403755868545
```

Confusion Matrix

```
[ ]  # Confusion Matrix
     model_conf = confusion_matrix(np.argmax(y_test, axis=1), np.argmax(y_pred, axis=1))
     print(model_conf)
```

```
[[313  10   3]
 [ 12  46   0]
 [  7   3  32]]
```

Model Evaluation

Classification Report

```
[ ]  # Confusion Matrix and Classification Report
     from sklearn.metrics import confusion_matrix, classification_report

     # Classification Report
     model_report = classification_report(np.argmax(y_test, axis=1), np.argmax(y_pred, axis=1))
     print(model_report)
```

```
                  precision    recall  f1-score   support

               0       0.94      0.96      0.95       326
               1       0.78      0.79      0.79        58
               2       0.91      0.76      0.83        42

        accuracy                           0.92       426
       macro avg       0.88      0.84      0.86       426
    weighted avg       0.92      0.92      0.92       426
```

Sains Data

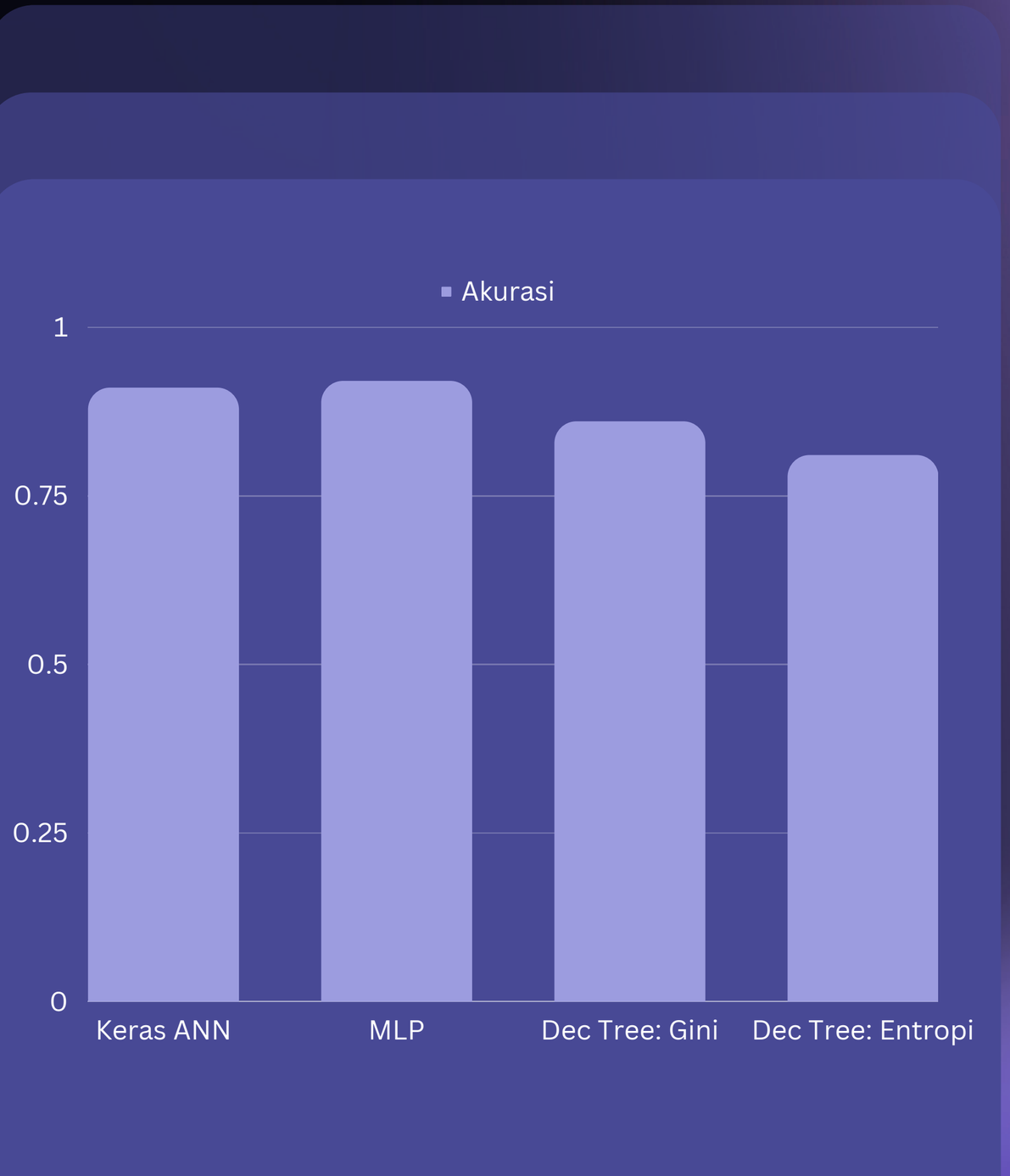
List of Content

Analisis Model

Akurasi Model

Setelah dilakukan berbagai tahapan yang ada termasuk *hyperparameter tuning* didapatkan hasil akhir akurasi sebagai berikut

Model Multilayer Perceptron	0.92	**Urutan ke-1**
Model Keras ANN	0.91	**Urutan ke-2**
Decision Tree: Gini	0.86	**Urutan ke-3**
Decision Tree: Entropi	0.81	**Urutan ke-4**



Analisis Model

F1 - Score

Setelah dilakukan berbagai tahapan yang ada termasuk *hyperparameter tuning* didapatkan hasil akhir f1-score sebagai berikut

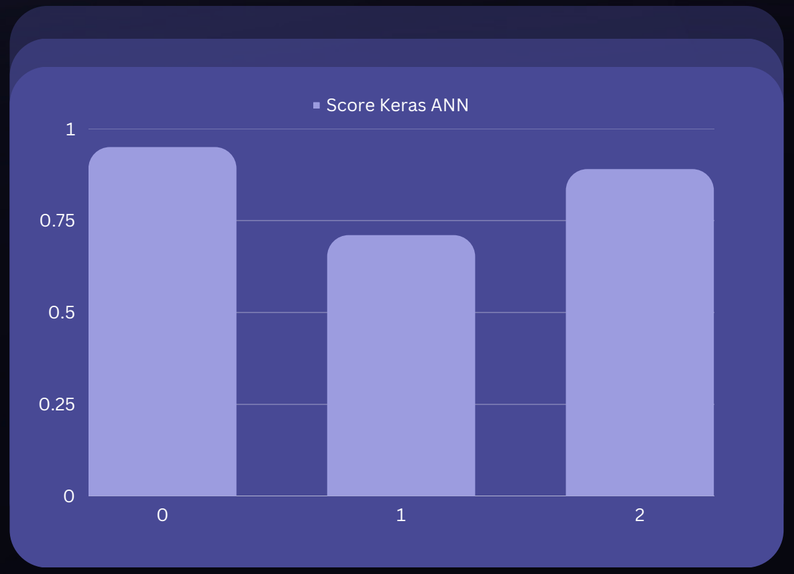
Model Multilayer Perceptron	0.92	**Urutan ke-1**
Model Keras ANN	0.91	**Urutan ke-2**
Decision Tree: Gini	0.87	**Urutan ke-3**
Decision Tree: Entropi	0.83	**Urutan ke-4**

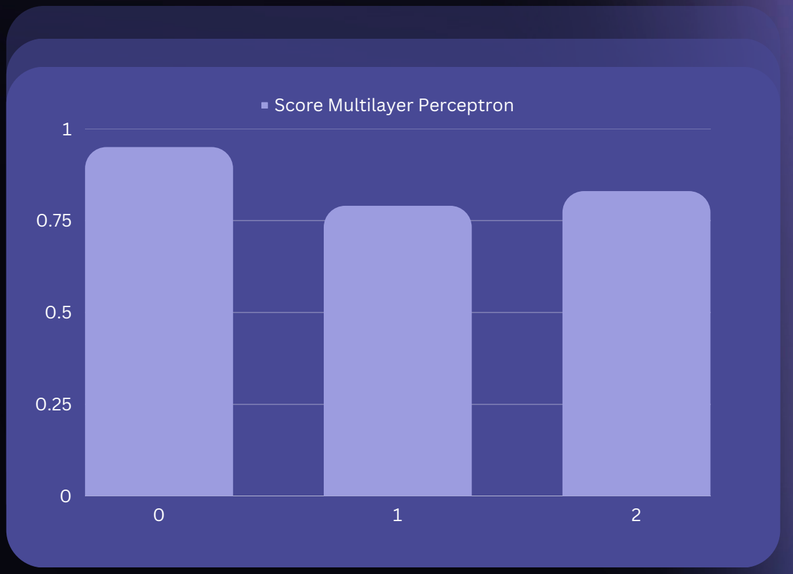
Analisis Model

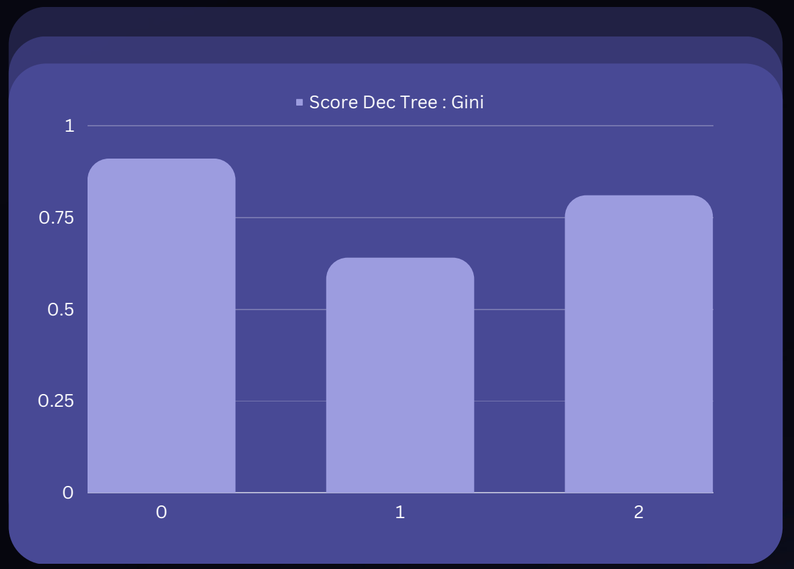
F1 - Score

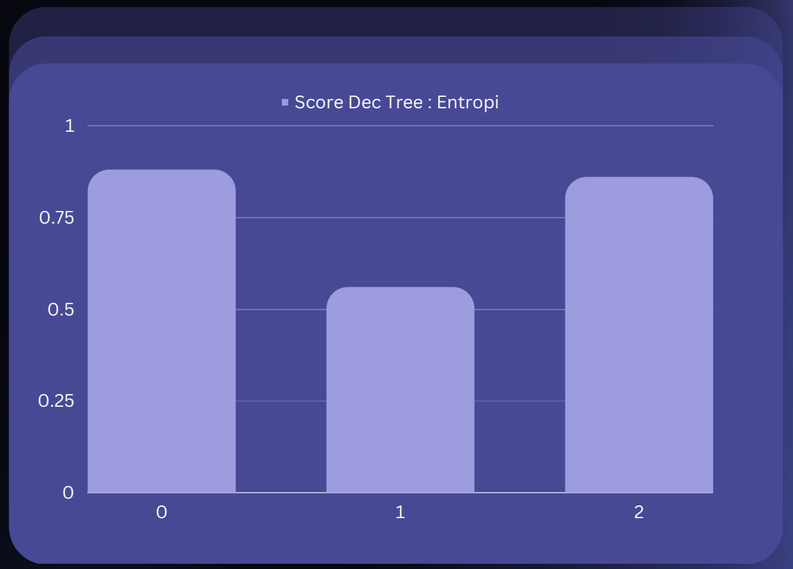
Berikut adalah visualisasi hasil f1-score untuk keempat metode dan seluruh kelas yang ada, yaitu

0	Normal
1	Suspect
2	Pathologic









Analisis Model

Features Important

Berikut adalah *code* yang dipakai untuk mencari *Features Important* dari ketiga model yang dipakai.

```
[ ]  import numpy as np
     from sklearn.neural_network import MLPClassifier

     # X dalam bentuk array numpy 2D
     X = np.array(X)

     # y dalam bentuk array numpy 1D
     y = np.array(y)

     model = MLPClassifier(hidden_layer_sizes=(10, 10))  # Ganti dengan konfigurasi MLP Anda
     model.fit(X, y)

     # Menghitung baseline accuracy
     baseline_accuracy = model.score(X, y)

     # Menghitung feature importance menggunakan perturbation-based
     feature_importance = np.zeros(X.shape[1])
     for i in range(X.shape[1]):
         X_permuted = X.copy()
         X_permuted[:, i] = np.random.permutation(X_permuted[:, i])
         permuted_accuracy = model.score(X_permuted, y)
         feature_importance[i] = baseline_accuracy - permuted_accuracy

     # Menormalisasi feature importance
     feature_importance /= feature_importance.max()

     # Mengurutkan fitur berdasarkan tingkat pentingnya
     sorted_indices = np.argsort(feature_importance)[::-1]

     # Menampilkan hasil
     for idx in sorted_indices:
         print(f'Fitur-{idx + 1}: {feature_importance[idx]}')
```

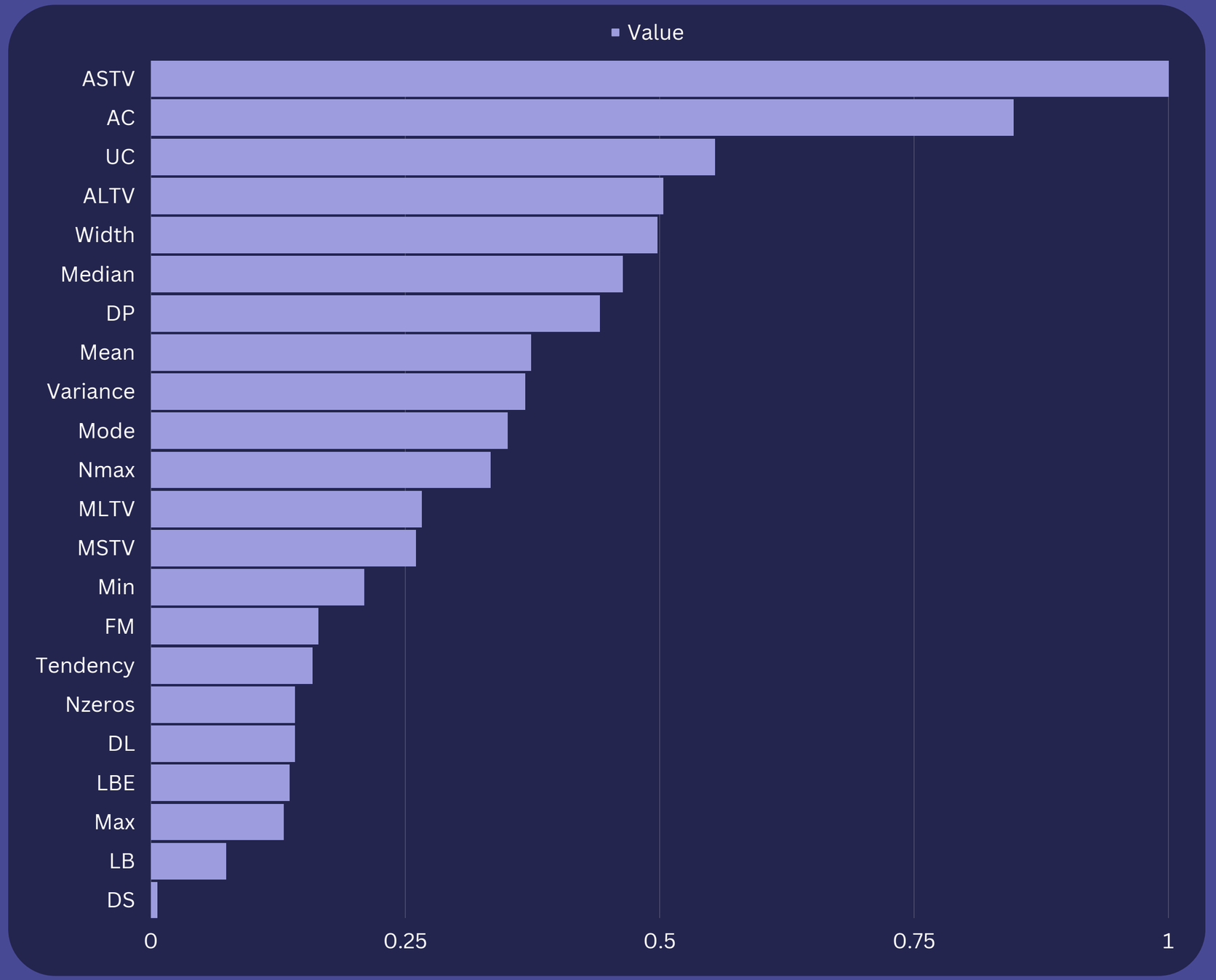
Analisis Model

Features Important

Model Multilayer Perceptron

KIta dapatkan 3 fitur yang paling penting dalam masalah klasifikasi ini adalah :

1. ASTV
2. AC
3. UC



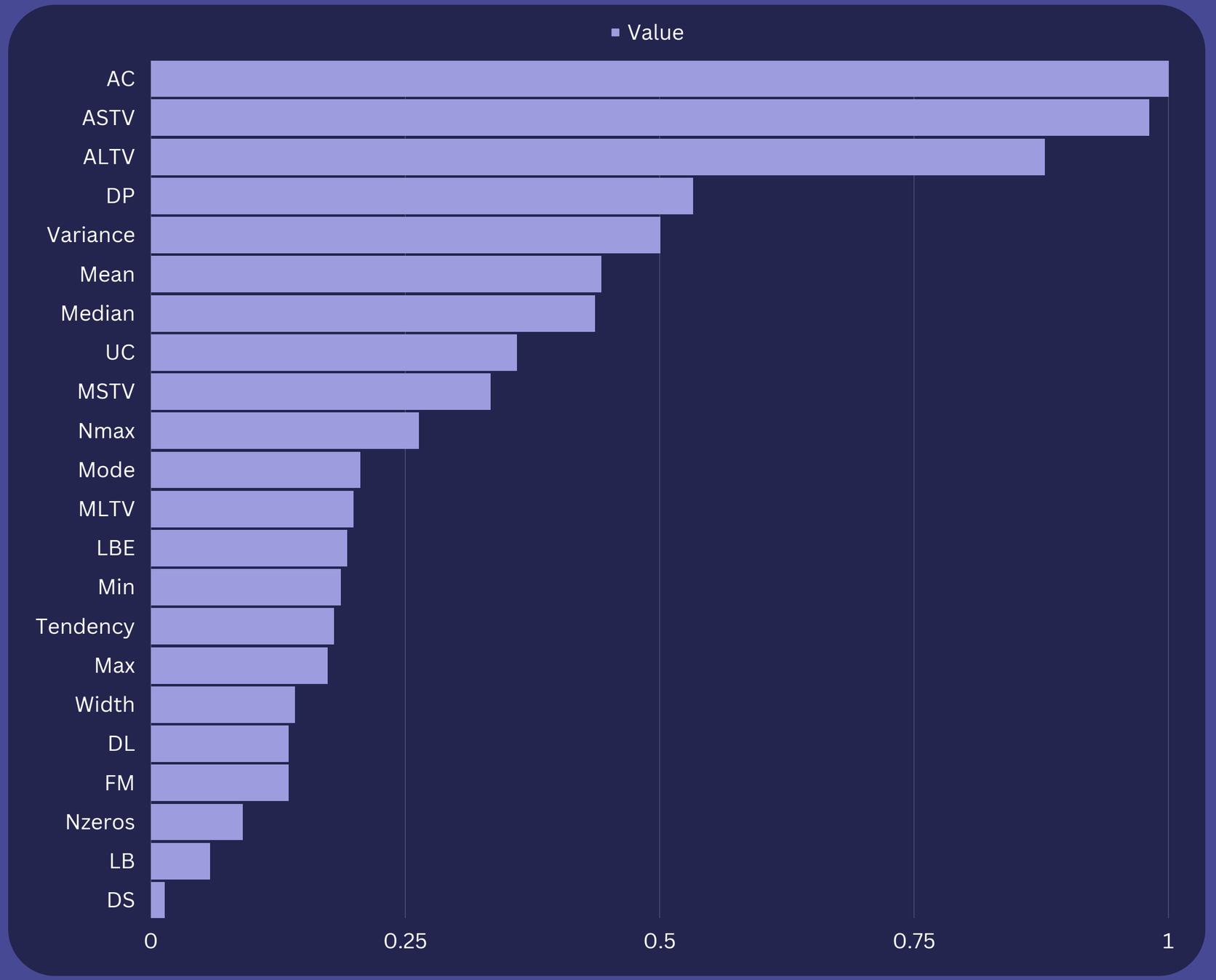
Analisis Model

Features Important

Model Keras ANN

KIta dapatkan 3 fitur yang paling penting dalam masalah klasifikasi ini adalah :

1. AC
2. ASTV
3. ALTV



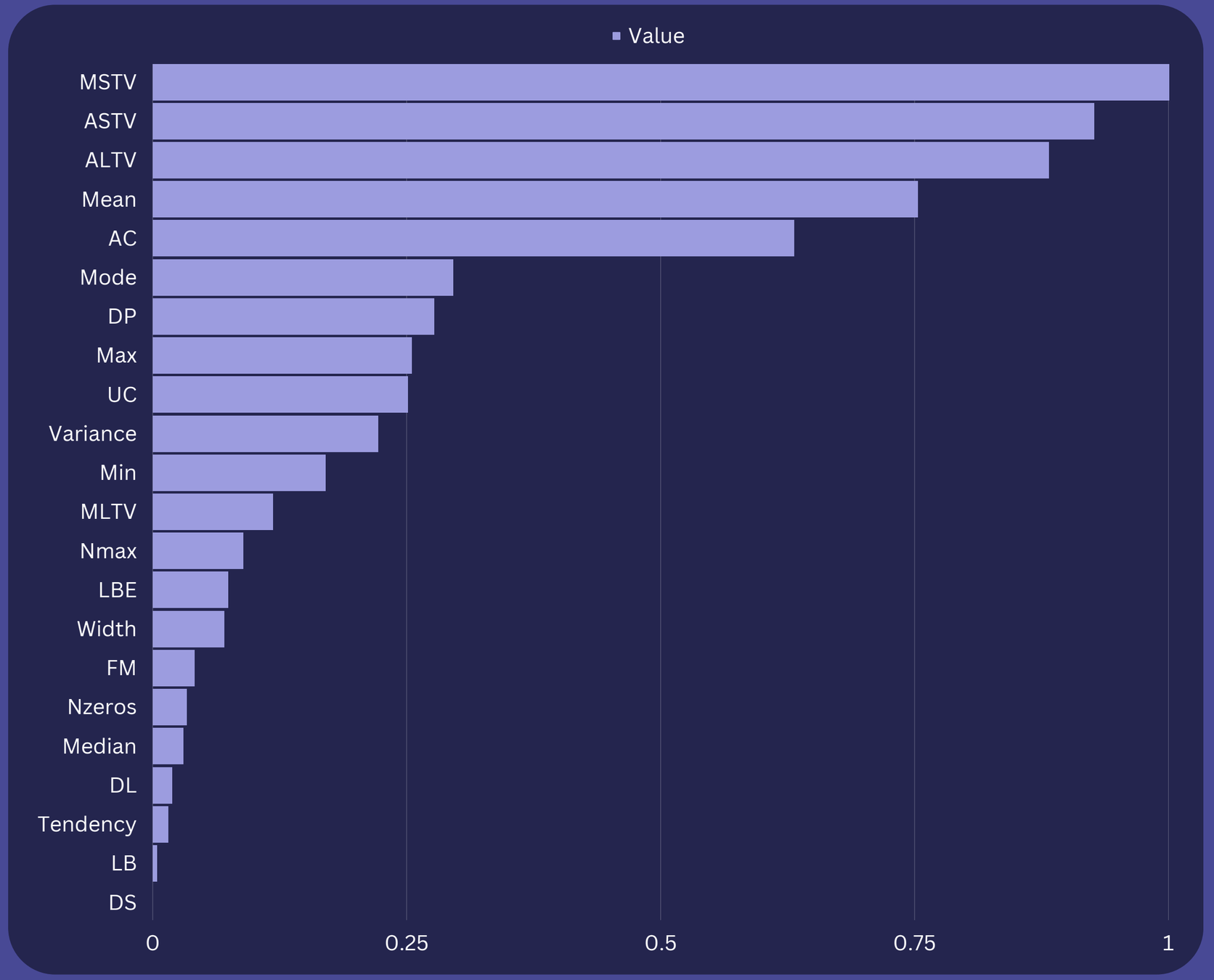
Analisis Model

Features Important

Model Decision Tree : Kriteria Gini

KIta dapatkan 3 fitur yang paling penting dalam masalah klasifikasi ini adalah :

1. MSTV
2. ASTV
3. ALTV



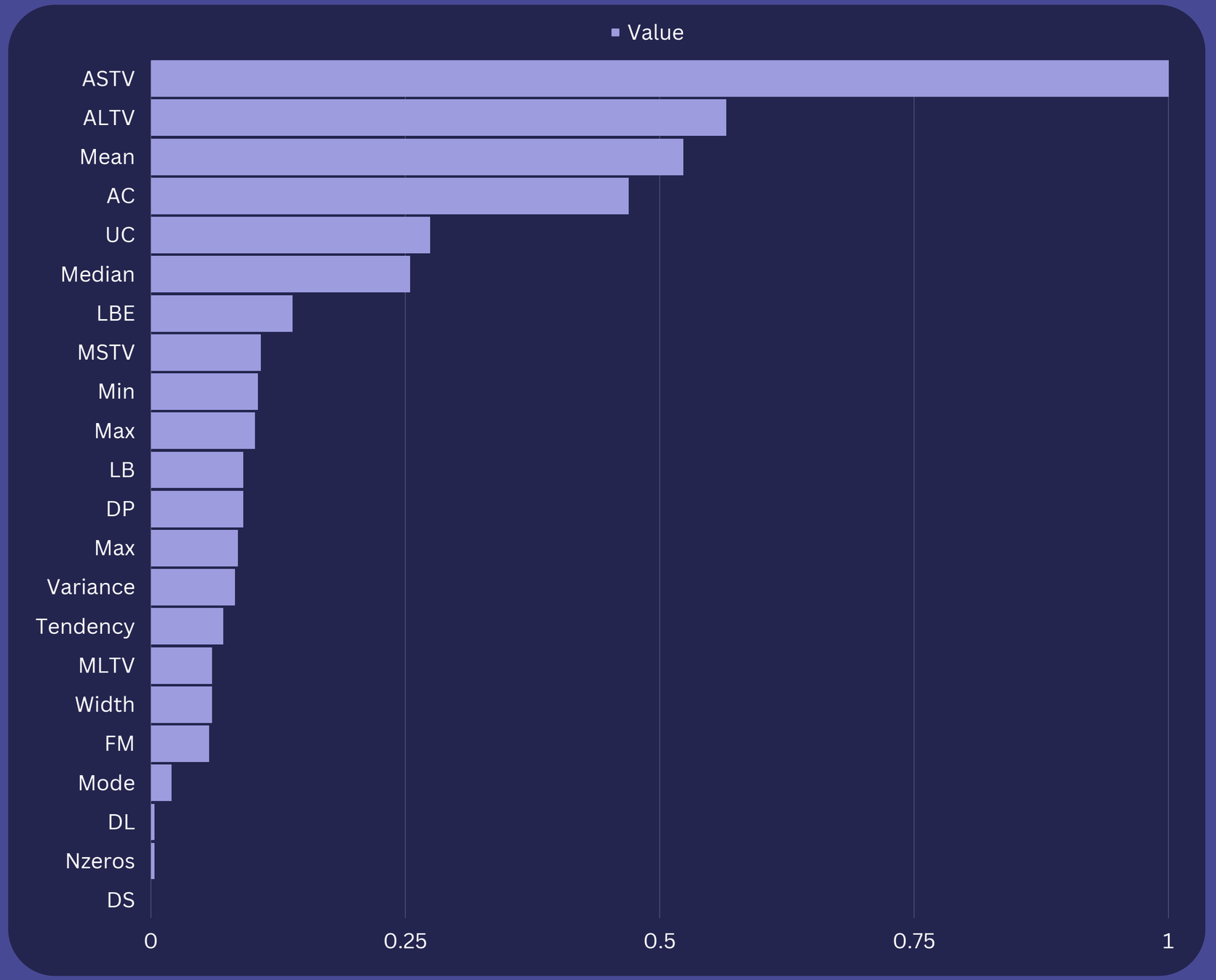
Analisis Model

Features Important

Model Decision Tree : Kriteria Entropi

KIta dapatkan 3 fitur yang paling penting dalam masalah klasifikasi ini adalah :

1. ASTV
2. ALTV
3. Mean

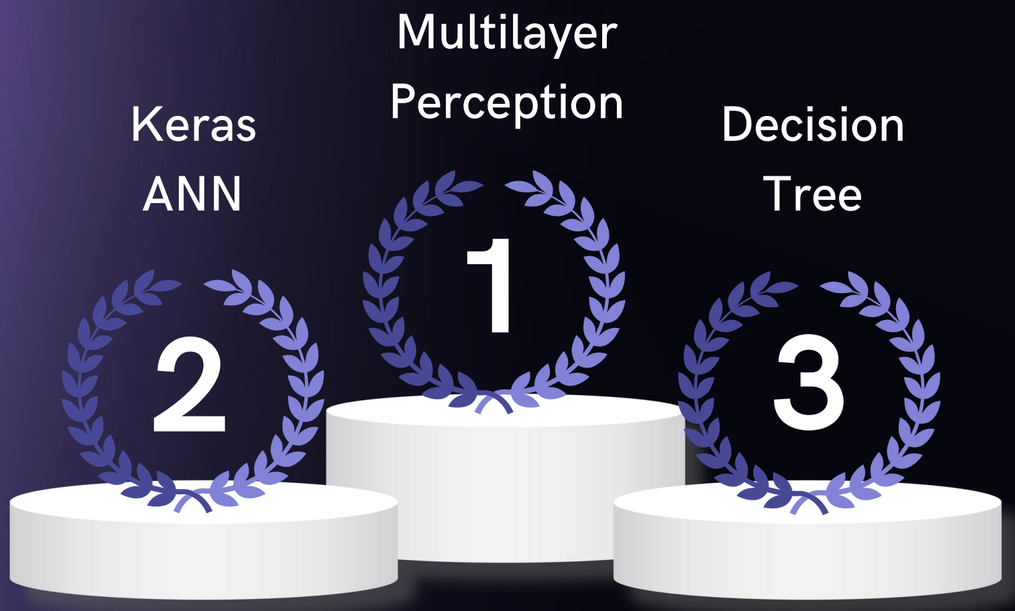


Sains Data

List of Content

Kesimpulan

Model Terbaik

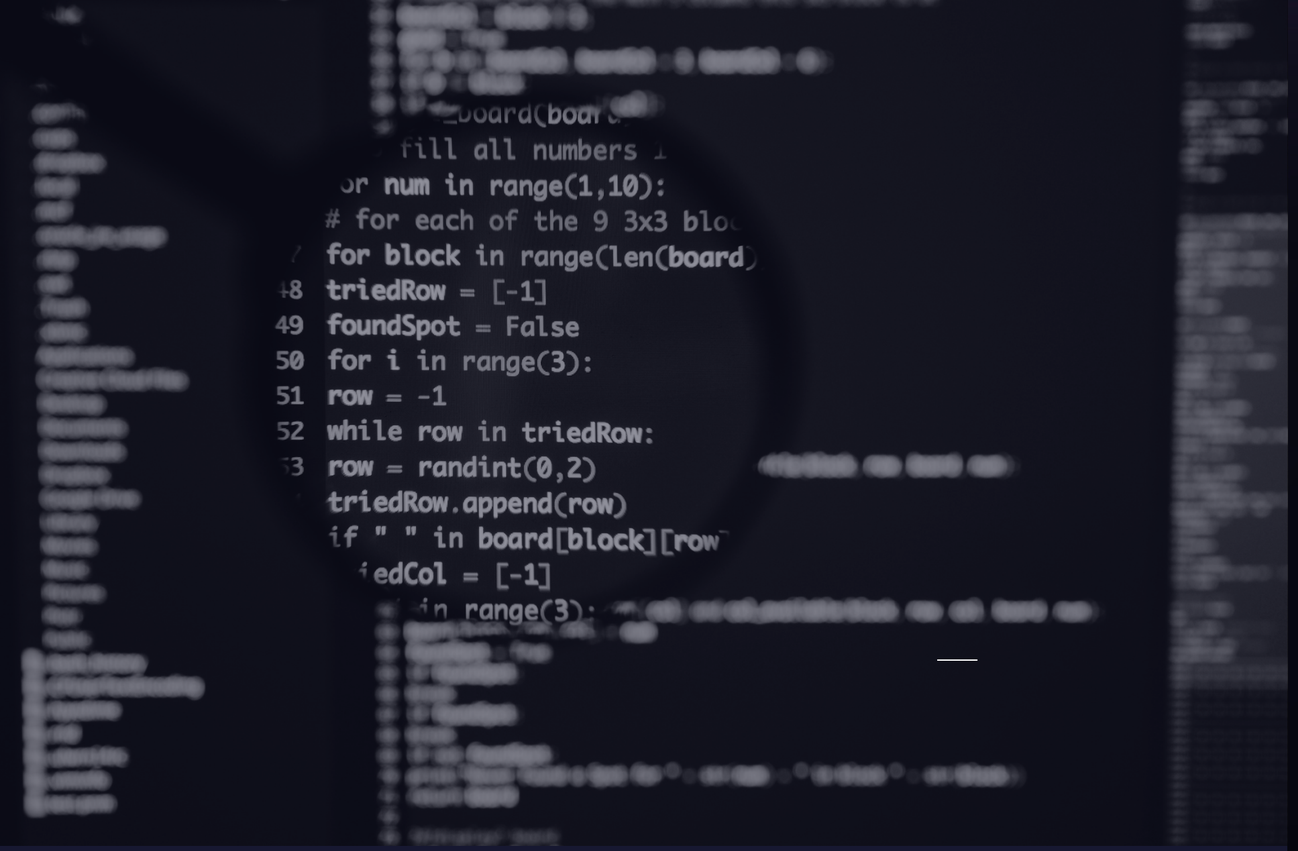


Fitur Terpenting

Kesimpulan Model Data

Penerapan model Keras, MLP, Decision Tree untuk memprediksi klasifikasi pada kasus *Fetal Cardiotography* sudah memberikan hasil yang relatif baik dengan akurasi 80% - 92%. Dengan akurasi model terbaik adalah **MLP** sebesar 92%.

Kemudian pada seluruh fitur pada dataset, didapatkan tiga fitur yang terpenting atau yang paling mempengaruhi dari masalah klasifikasi *Fetal Cardiotography* yaitu, ASTV, ALTV, dan AC.



Saran

Berdasarkan semua tahapan yang sudah kami lakukan, kami mempunyai saran untuk menjadikannya lebih baik.

Dalam tahapan data cleaning dapat dilakukan pengecekan outlier dan penghapusan outlier, namun untuk proses imputasinya, diperlukan lagi berbagai metode untuk mengeksekusi handling missing value.

Sains Data

Terima Kasih

...

Kelompok 4