

## ANGGOTA KELOMPOK



AFDATUL CHOFIDAH (222111848)



DINA YANTI NAINGGOLAN (222111992)



ARZUDA QOLBIN MULYA (222111926)



SETYA HADI NUGROHO ( 222112358 )



# **Fetal Health Classification**

**Class: Normal, Suspect, Pathological**  
**Menggunakan Cardiotocograms (CTGs)**  
**dengan jumlah record 2126 janin**

sumber: Ayres de Campos et al. (2000) SisPorto 2.0 A Program for Automated Analysis of Cardiotocograms. J Matern Fetal Med 5:311-318

# Definisi Variabel

**baseline\_value** berisi data denyut jantung janin atau FHR (Fetal Health Rate) .

**accelerations** berisi data jumlah akslerasi janin per detik.

**fetal\_movement** berisi data jumlah gerakan janin per detik.

**uterine\_contractions** berisi data jumlah konstraksi uterus per detik.

**light\_decelerations** berisi data jumlah penurunan detak jantung (ringan) per detik.

**severe\_decelerations** berisi data jumlah penurunan detak jantung (parah) per detik.

**prolongued\_decelerations** berisi data jumlah penurunan detak jantung (berkepanjangan) per detik.

**Abnormal\_short\_term\_variability** berisi data persentase waktu dengan variabilitas jangka Panjang yang abnormal.

# K-Nearest Neighbors



01

Split Dataset  
untuk Data  
Training dan  
Data Testing



02

Pembentukan  
Model



03

Confusion  
Matrix dan  
Binary  
Confusion  
Matrix



04

Hasil Evaluasi  
Model

# K-Nearest Neighbors

- **Pengertian**

K-nearest neighbors (KNN) adalah sebuah algoritma dalam machine learning yang digunakan untuk klasifikasi dan regresi. Prinsip utama dari KNN adalah dengan mencari kelas atau nilai target dari data baru berdasarkan mayoritas dari kelas atau nilai target tetangga terdekatnya

- **Tujuan**

Tujuan dari KNN adalah untuk memprediksi label atau nilai target dari sebuah data baru berdasarkan data yang telah ada dalam dataset, dengan mengasumsikan bahwa data-data yang memiliki atribut atau fitur yang serupa cenderung memiliki label atau nilai target yang sama.



# Split Data

80% Training, 20% Testing

```
# Split dataframe untuk data training dan data testing  
# Data training 80%, data testing 20% dari record  
from sklearn.model_selection import train_test_split  
df_training, df_testing = train_test_split(data, test_size=0.2)
```

# Pembentukan Model

```
# Pembentukan Model
from sklearn.neighbors import KNeighborsClassifier
X_train = df_training.iloc[:, 0:11]
y_train = df_training.iloc[:, 11]

nn3_euclidean = KNeighborsClassifier(n_neighbors=3, metric="euclidean")
nn3_euclidean.fit(X_train, y_train)

# Testing
X_test = df_testing.iloc[:, 0:11]
y_test = df_testing.iloc[:, 11]
y_pred = nn3_euclidean.predict(X_test)
```



# Confusion Matrix

```
# Rekap Hasil Testing dengan Confusion Matrix
from sklearn.metrics import confusion_matrix

CM = confusion_matrix(y_test, y_pred)
CM

array([[329,   9,   2],
       [ 23,  27,   1],
       [  9,   3,  23]])

import numpy as np
class_metrics = []

# Loop untuk setiap kelas
for class_index in range(3):
    binary_conf_matrix = np.zeros((2, 2), dtype=int)

    # Mengisi elemen tiap matriks biner
    for i in range(3):
        for j in range(3):
            if i == j == class_index:
                binary_conf_matrix[0, 0] += CM[i, j] # TP
            elif i == class_index and j != class_index:
                binary_conf_matrix[0, 1] += CM[i, j] # FP
            elif i != class_index and j == class_index:
                binary_conf_matrix[1, 0] += CM[i, j] # FN
            else:
                binary_conf_matrix[1, 1] += CM[i, j] # TN

    class_metrics.append(binary_conf_matrix)
```

Binary CM saat Class 1 = True:  
[[329 11]  
 [ 32 54]]

(TP): 329  
(TN): 54  
(FP): 11  
(FN): 32

Binary CM saat Class 2 = True:  
[[ 27 24]  
 [ 12 363]]

(TP): 27  
(TN): 363  
(FP): 24  
(FN): 12

Binary CM saat Class 3 = True:  
[[ 23 12]  
 [ 3 388]]

(TP): 23  
(TN): 388  
(FP): 12  
(FN): 3

# Evaluasi Model

```
for i in range(3):
    Accuracy.append((TP[i]+TN[i]) / (TP[i]+TN[i]+FP[i]+FN[i]))
    Precision.append(TP[i] / (TP[i]+FP[i]))
    Recall.append(TP[i] / (TP[i]+FN[i]))
    F_1Scores.append((2*Precision[i]*Recall[i]) / (Precision[i]+Recall[i]))
```

```
# Nilai Evaluasi Keseluruhan
# F1-Score Makro = Rata2 dari F-1Score
Macro_F1_Score = np.mean(F_1Scores)
print(f"Nilai Makro F-1Score: {Macro_F1_Score}")

# Weighted F1
# Pembobotan nilai F1 berdasarkan jumlah klasifikasi class pada data Testing
sum_class = np.sum(CM, axis=1)
weighted_f1 = np.dot(sum_class, F_1Scores) / np.sum(sum_class)
print(f"Nilai Weighted F-1Score: {weighted_f1}")
```

```
Nilai Makro F-1Score: 0.7642524730478707
Nilai Weighted F-1Score: 0.8829519307718972
```

```
[ Nilai Evaluasi untuk Class 1 = True ]
Akurasi: 0.8990610328638498
Presisi: 0.9676470588235294
Recall: 0.9113573407202216
F1-Scores: 0.9386590584878745
```

```
[ Nilai Evaluasi untuk Class 2 = True ]
Akurasi: 0.9154929577464789
Presisi: 0.5294117647058824
Recall: 0.6923076923076923
F1-Scores: 0.5999999999999999
```

```
[ Nilai Evaluasi untuk Class 3 = True ]
Akurasi: 0.9647887323943662
Presisi: 0.6571428571428571
Recall: 0.8846153846153846
F1-Scores: 0.7540983606557377
```

# Evaluasi Model

```
[ ] from sklearn.metrics import accuracy_score,precision_score, recall_score

# Menghitung akurasi
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Menghitung presisi
precision = precision_score(y_test, y_pred, average='weighted')
print("Precision:", precision)

# Menghitung recall
recall = recall_score(y_test, y_pred, average='weighted')
print("Recall:", recall)

Accuracy: 0.9014084507042254
Precision: 0.9005144446518645
Recall: 0.9014084507042254
```

# Decision Tree



01

Split Dataset  
untuk Data  
Training dan  
Data Testing



02

Pembentukan  
Model



03

Confusion  
Matrix



04

Hasil Evaluasi  
Model

# Decision Tree

- **Pengertian**

Decision tree adalah sebuah metode dalam machine learning yang digunakan untuk memecah data menjadi potongan-potongan yang lebih kecil berdasarkan serangkaian keputusan berbasis aturan-aturan

- **Tujuan**

Tujuannya adalah untuk membangun model prediktif yang dapat mengambil keputusan atau membuat prediksi dengan mengikuti alur dari pohon keputusan, di mana setiap simpul dalam pohon mewakili suatu keputusan berdasarkan fitur-fitur dari data, dan setiap cabang menunjukkan kemungkinan hasil dari keputusan tersebut.

# Decision Tree

```
library(haven)
library(readr)
data <- read.csv("D:/STIS/SI/SMT 6/DATMIN/UTS/P4/janin.csv")
head(data)

#transformasi data (Mengubah nilai numerik menjadi kategori)
# Tentukan jumlah kategori yang diinginkan
num_bins <- 3

# Loop melalui setiap kolom numerik dan transformasikan menjadi kategori
for (col in names(data)) {
  if (is.numeric(data[[col]])) {
    data[[col]] <- cut(data[[col]], breaks = num_bins, labels = FALSE)
    data[[col]] <- as.factor(data[[col]])
  }
}

# 1 = Normal
# 2 = Suspect
# 3 = Pathological
```



# Split Data

## 80% Training, 20% Testing

```
# Pisahkan dataset menjadi data training dan data testing
set.seed(42) # Untuk hasil yang dapat direproduksi
Sample <- sample(c(TRUE, FALSE), nrow(data), replace=TRUE,
prob=c(0.8,0.2))
testing <- data[Sample, ]
learning <- data[-Sample, ]
```

# Pembentukan Model

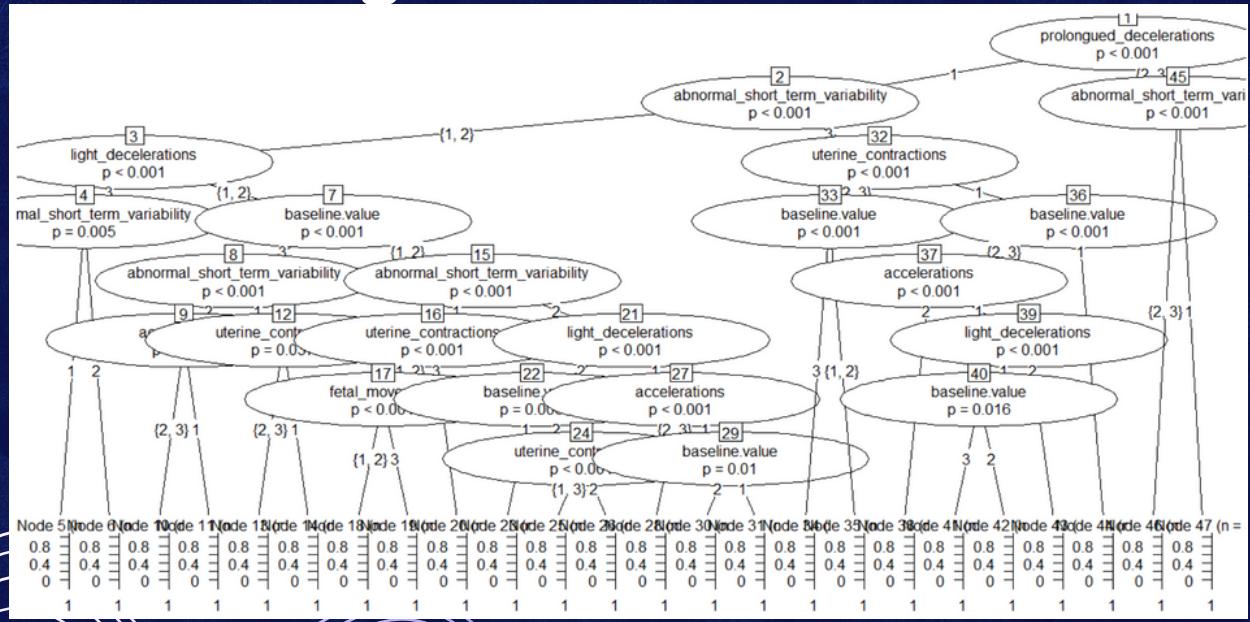
Dengan fungsi rpart

```
# Menampilkan model tree yang terbentuk
form <- as.formula(fetal_health ~ baseline.value + accelerations + fetal_movement + uterine_contractions +
+ light_decelerations + severe_decelerations + prolonged_decelerations +
abnormal_short_term_variability)
tree.1 <- rpart(form,data = learning, control = rpart.control(minsplit = 20,cp=0))
```

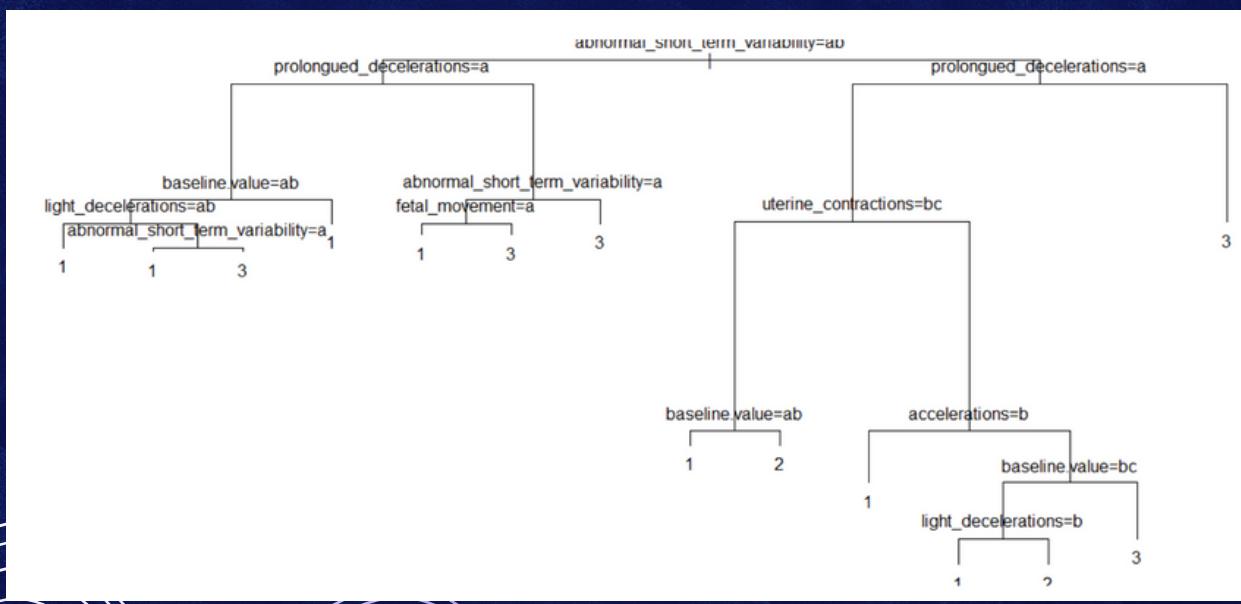
Dengan fungsi ctree

```
output.tree <- ctree(fetal_health ~ baseline.value + accelerations + fetal_movement +
uterine_contractions + light_decelerations + severe_decelerations + prolonged_decelerations +
abnormal_short_term_variability, data=learning)
```

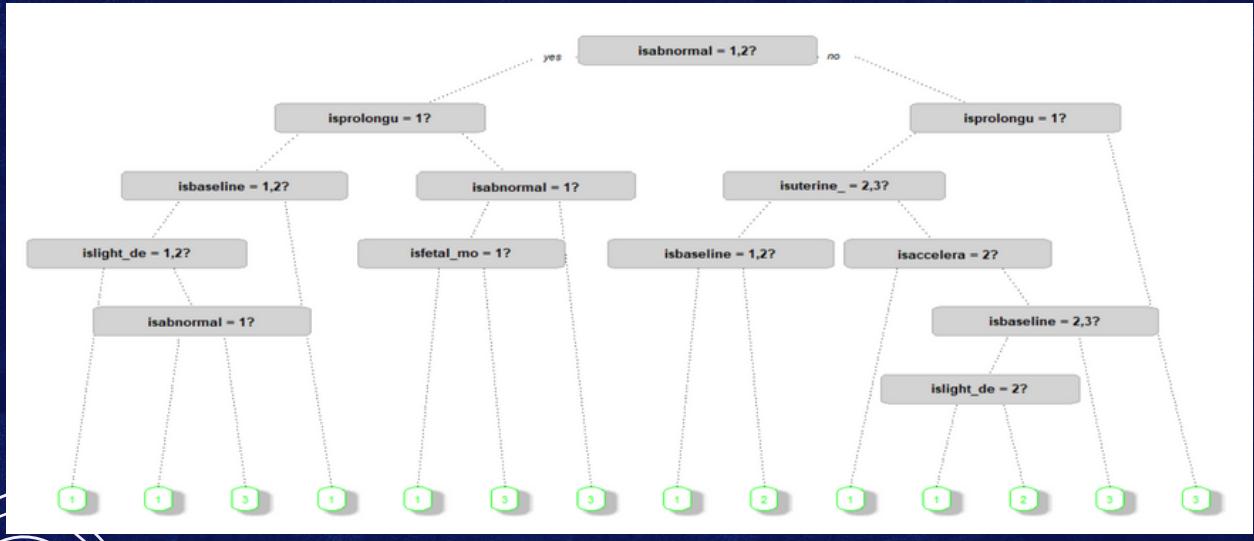
# Model dengan Ctree



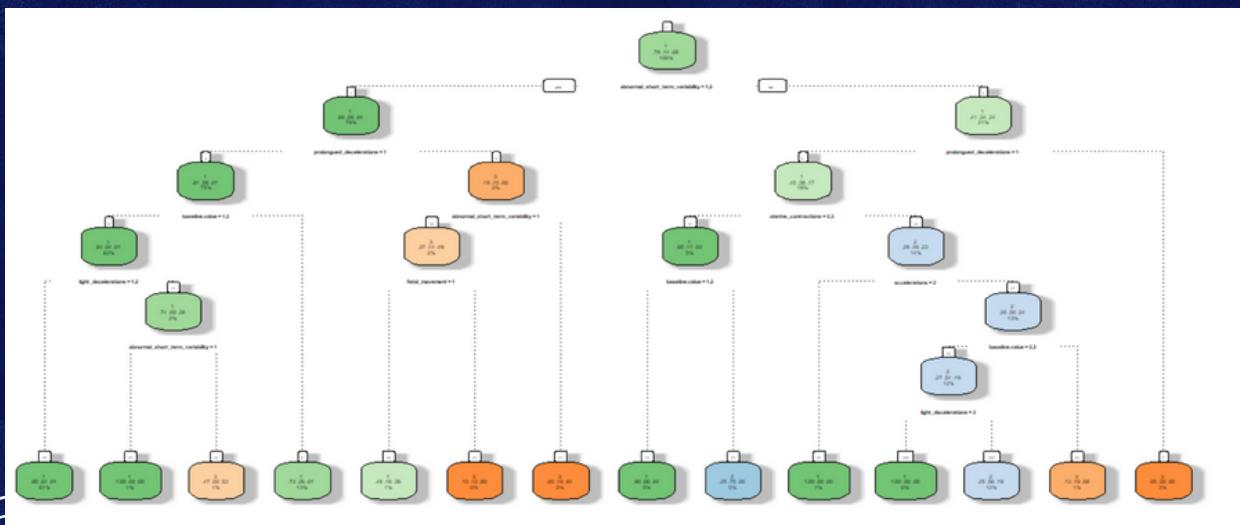
# Model dengan rpart



# Model prp



# Model fancyRpartPlot



# Confusion Matrix

```
# Confusion Matrix  
CM <- table(testing$fetal_health, Prediksi)  
CM  
````
```

| Prediksi | 1    | 2   | 3  |
|----------|------|-----|----|
| 1        | 1250 | 55  | 20 |
| 2        | 115  | 111 | 10 |
| 3        | 15   | 39  | 88 |

- Terdapat 1250 data janin yang diprediksi sebagai Normal (True Positive).

- Terdapat 111 data janin yang diprediksi sebagai suspect (True Positive).

- Terdapat 88 data janin yang diprediksi sebagai Pathological (True Positive).

# Evaluasi Model

```
# Akurasi  
accuracy <- (sum(diag(CM)))/sum(CM)  
  
print("Accuracy:")  
print(accuracy)
```

```
[1] "Accuracy:"  
[1] 0.8508514
```

```
# Precision  
precision <- diag(CM) / colSums(CM)  
  
print("Precisions for Each Class:")  
print(precision)
```

```
[1] "Precisions for Each Class:"  
     1      2      3  
0.9057971 0.5414634 0.7457627
```

```
# F1 Score  
precision <- diag(CM) / colSums(CM)  
recall <- diag(CM) / rowSums(CM)  
F1_score <- 2 * (precision * recall) / (precision + recall)  
  
print("F1-score for Each Class:")  
print(F1_score)
```

```
[1] "F1-score for Each Class:"  
     1      2      3  
0.9242144 0.5034014 0.6769231
```

# Evaluasi Model

```
# AUC (Area Under the ROC Curve)
calculate_auc <- function(confusion_matrix) {
  # Inisialisasi vektor untuk menyimpan AUC untuk setiap kelas
  aucs <- numeric(nrow(confusion_matrix))

  for (i in 1:nrow(confusion_matrix)) {
    # Menghitung true positive rate (sensitivity) dan false positive rate untuk kelas i
    true_positive_rate <- confusion_matrix[i, i] / sum(confusion_matrix[i, ])
    false_positive_rate <- sum(confusion_matrix[-i, i]) / sum(confusion_matrix[-i, ])

    # Menghitung AUC untuk kelas i menggunakan trapezoidal rule
    aucs[i] <- (true_positive_rate + false_positive_rate) / 2
  }

  return(aucs)
}

print("AUC for Each Class:")
print(aucs)
```

```
[1] "AUC for Each Class:"
[1] 0.4702157 0.4520649 0.4760631
```

# Linear Discriminant Analysis



**01**

Split Dataset  
untuk Data  
Training dan  
Data Testing



**02**

Pembentukan  
Model



**03**

Confusion  
Matrix



**04**

Hasil Evaluasi  
Model

# LDA ( Linear Discriminant Analysis )

- **Pengertian**

Linear Discriminant Analysis (LDA) adalah metode statistik yang digunakan untuk menemukan kombinasi linear dari fitur-fitur yang paling baik memisahkan atau membedakan antara dua atau lebih kelas dalam sebuah dataset.

- **Tujuan**

Tujuannya adalah untuk mencari proyeksi linier dari data ke dimensi yang lebih rendah sehingga kelas-kelas dapat dipisahkan sejauh mungkin. LDA sering digunakan untuk reduksi dimensi dalam klasifikasi, pemrosesan sinyal, pengolahan citra, dan pengenalan pola.

# Split Data

## 80% Training, 20% Testing



```
set.seed(123)
train_index <- sample(seq(nrow(datalda)),
                      size = floor(0.8 * nrow(datalda)), replace = F)
training_data <- datalda[train_index, ]
test_data <- datalda[-train_index, ]
```

# Pembentukan Model data training

```
Call:  
lda(fetal_health ~ ., data = training_data_clean)  
  
Prior probabilities of groups:  
1 2 3  
0.78235294 0.13882353 0.07882353
```

Membentuk model berdasarkan data training.

# Fungsi diskriminan

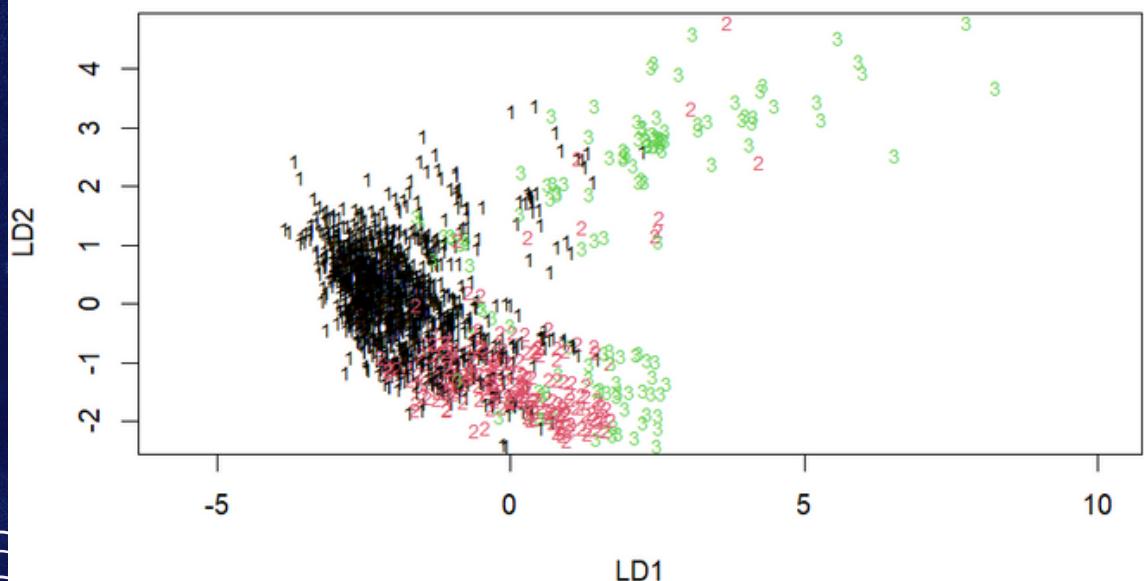
Coefficients of linear discriminants:

|                                                        | LD1            | LD2            |
|--------------------------------------------------------|----------------|----------------|
| `baseline_value`                                       | 0.004175366    | -0.053266808   |
| accelerations                                          | -40.246379959  | 91.130256923   |
| fetal_movement                                         | 0.162169777    | -2.422974220   |
| uterine_contractions                                   | -74.168825634  | 61.397741422   |
| light_decelerations                                    | 40.147372177   | 71.887326051   |
| prolongued_decelerations                               | 1794.140402529 | 1025.758629759 |
| abnormal_short_term_variability                        | 0.031616498    | 0.003388000    |
| mean_value_of_short_term_variability                   | 0.128501923    | 0.172122030    |
| percentage_of_time_with_abnormal_long_term_variability | 0.038058243    | -0.008017409   |
| mean_value_of_long_term_variability                    | 0.008845206    | -0.002631377   |

## Proportion of trace

```
Proportion of trace:  
LD1      LD2  
0.8157  0.1843
```

- LD1 menjelaskan sekitar 81.57% dari total variabilitas dalam data
- LD2 menjelaskan sekitar 18.43% dari total variabilitas dalam data
- Semakin tinggi proporsi trace, semakin besar kontribusi LD terhadap pemisahan antar grup dalam data.
- LD1 memiliki kontribusi yang lebih besar daripada LD2 dalam menjelaskan variasi dalam data.



# Melakukan prediksi pada data testing

```
```{r}
predicted <- predict(object = linearDA, newdata = test_data)
table(actual = test_data$fetal_health, predicted = predicted$class)
```



	predicted		
actual	1	2	3
1	307	12	6
2	24	33	2
3	7	12	23


```

# Evaluasi model

```
```{r}
# Akurasi
accuracy <- (sum(diag(mat))/sum(mat))
accuracy
````
```

```
[1] 0.8521127
```

```
```{r}
#Precision
precision <- diag(mat)/colSums(mat)
precision
````
```

```
1       2       3
0.9082840 0.5789474 0.7419355
```

```
```{r}
#F1 store
precision <- diag(mat)/colSums(mat)
recall <- diag(mat)/rowSums(mat)
F1_score <- 2 * (precision *recall)/(precision +recall)
F1_score
````
```

```
1       2       3
0.9260935 0.5689655 0.6301370
```

```
"AUC for Each Class:"
0.6257730 0.3123586 0.2842262
```

# Kesimpulan

**F1- Score Tertinggi = 0.926.. ( Linear Discriminant Analysis )**

**Presisi Tertinggi = 0.908.. ( Linear Discriminant Analysis )**

**Akurasi Tertinggi = 0,901.. (K Nearest Neighbors )**

# Thank you very much!

