BAB IV
HASIL PENELITIAN DAN PEMBAHASAN

Pada bab ini akan dijabarkan terkait hasil-hasil yang telah didapatkan dari hasil klasifikasi menggunakan metode C4.5. dan *Random Forest* serta C4.5 dengan seleksi fitur dan *Random Forest* dengan seleksi fitur

4.1 Tahap Pengujian

Pada tahap awal pengujian kita melakukan import library terlebih dahulu pada google colaboratory dengan mengetikan perintah sebagai berikut :

```
# For Data Processing
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
import plotly.express as px

# Import ML Model Classifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier


# For Model Evaluation
from sklearn.metrics import accuracy_score, classification_report
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import confusion_matrix
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import roc_curve, auc
from matplotlib import pyplot
```

4.1.1 Dataset

Kumpulan data dikumpulkan dari pembelajaran mesin UCI repositori [27]. Dataset berisi 400 instance dengan 24 atribut dan 1 atribut kelas. Atribut-atribut ini disajikan pada tabel berikut. Dataset berisi 400 instance (250 CKD, 150 notCKD) dan jumlah Atribut: 24 + kelas = 25 (11 numerikal, 14 nominal).

Dataset dapat kita import dengan perintah sebagai berikut :

```
df = pd.read_csv('kidney_disease.csv')
```

Data tersebut dapat dilihat dengan mengetikan perintah :

```
df.info()
```

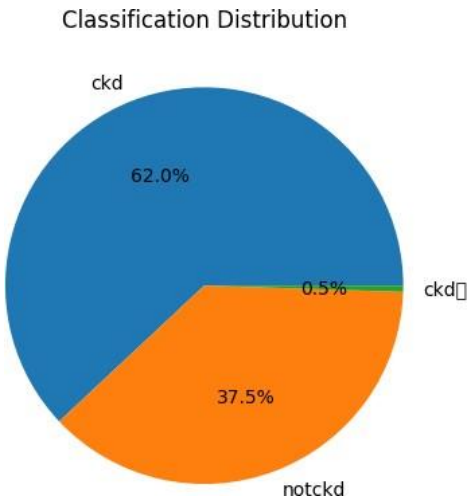
Hasil yang diperoleh adalah sebagai berikut :

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 25 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   age                     391 non-null    float64
 1   blood_pressure          388 non-null    float64
 2   specific_gravity        353 non-null    float64
 3   albumin                 354 non-null    float64
 4   sugar                   351 non-null    float64
 5   red_blood_cells         248 non-null    object
 6   pus_cell                335 non-null    object
 7   pus_cell_clumps         396 non-null    object
 8   bacteria                396 non-null    object
 9   blood_glucose_random    356 non-null    float64
 10  blood_urea              381 non-null    float64
 11  serum_creatinine        383 non-null    float64
 12  sodium                  313 non-null    float64
 13  potassium               312 non-null    float64
 14  haemoglobin             348 non-null    float64
 15  packed_cell_volume      330 non-null    object
 16  white_blood_cell_count  295 non-null    object
 17  red_blood_cell_count    270 non-null    object
 18  hypertension            398 non-null    object
 19  diabetes_mellitus       398 non-null    object
 20  coronary_artery_disease 398 non-null    object
 21  appetite                399 non-null    object
 22  peda_edema              399 non-null    object
 23  aanemia                 399 non-null    object
 24  class                   400 non-null    object
dtypes: float64(11), object(14)
```

Melihat distribusi data, dengan mengetikan perintah sebagai berikut :

```
# Checking target class
plt.pie(df['class'].value_counts(), labels=df['class'].value_counts().index, autopct='%1.1f%%')
plt.title('Classification Distribution')
plt.show()
```

Hasil yang diperoleh 62% ckd(248), 37,5%(150) notckd dan 0,5%(2) missing value, Secara detail distribusi data *classification* awal dapat dilihat pada gambar 4.1.



Gambar 4 1 Distribusi Data *Classification* Awal

4.1.2 Data Preparation

Fase ini merupakan proses penyiapan data yang akan di olah dan semua data harus lengkap dengan variable dan datanya. Data diambil dari kaggle https://www.kaggle.com/datasets/mansoordaku/ckdisease). Data yang sudah siap ckdisease.csv diolah menggunakan tool google colaboratory. Pada tahap ini dilakukan untuk menangani data-data yang tidak diperlukan, seperti data yang memiliki missing value. Data yang memiliki missing value nantinya akan dilakukan penanganan dengan cara menghapus melakukan rata-rata dan subtitusi.

Beberapa langkah dilakukan dalam data preparation:

1) Mencetak nilai unik dalam data

```
# Printing unique values in data
for i in df.columns:
    print('unique values in "{}":\n'.format(i),df[i].unique())
```

Hasil yang diperoleh adalah sebagai berikut :

```
  2.9   1.7   3.6   5.6   6.5   4.4  10.2  11.5   0.5  12.2   5.3   9.2
 13.8  16.9   6.    7.1  18.    2.3  13.   48.1  14.2  16.4   2.6   7.5
  4.3  18.1  11.8   9.3   6.8  13.5  12.8  11.9  12.   13.4  15.2  13.3
  8.4 ]
unique values in "sodium":
 [  nan 111.  142.  104.  114.  131.  138.  135.  130.  141.  139.    4.5
 136.  129.  140.  132.  133.  134.  125.  163.  137.  128.  143.  127.
 146.  126.  122.  147.  124.  115.  145.  113.  120.  150.  144. ]
unique values in "potassium":
 [ nan  2.5  3.2  4.   3.7  4.2  5.8  3.4  6.4  4.9  4.1  4.3  5.2  3.8
  4.6  3.9  4.7  5.9  4.8  4.4  6.6 39.   5.5  5.   3.5  3.6  7.6  2.9
  4.5  5.7  5.4  5.3 47.   6.3  5.1  5.6  3.   2.8  2.7  6.5  3.3]
unique values in "haemoglobin":
 [15.4 11.3  9.6 11.2 11.6 12.2 12.4 10.8  9.5  9.4  9.7  9.8  5.6  7.6
 12.6 12.1 12.7 10.3  7.7 10.9  nan 11.1  9.9 12.5 12.9 10.1 12.  13.
  7.9  9.3 15.  10.   8.6 13.6 10.2 10.5  6.6 11.   7.5 15.6 15.2  4.8
  9.1  8.1 11.9 13.5  8.3  7.1 16.1 10.4  9.2  6.2 13.9 14.1  6.  11.8
 11.7 11.4 14.   8.2 13.2  6.1  8.  12.3  8.4 14.3  9.   8.7 10.6 13.1
 10.7  5.5  5.8  6.8  8.8  8.5 13.8 11.5  7.3 13.7 12.8 13.4  6.3  3.1
 17.  15.9 14.5 15.5 16.2 14.4 14.2 16.3 14.8 16.5 15.7 13.3 14.6 16.4
 16.9 16.  14.7 16.6 14.9 16.7 16.8 15.8 15.1 17.1 17.2 15.3 17.3 17.4
 17.7 17.8 17.5 17.6]
unique values in "packed_cell_volume":
 ['44' '38' '31' '32' '35' '39' '36' '33' '29' '28' nan '16' '24' '37' '30'
 '34' '40' '45' '27' '48' '\t?' '52' '14' '22' '18' '42' '17' '46' '23'
 '19' '25' '41' '26' '15' '21' '43' '20' '\t43' '47' '9' '49' '50' '53'
 '51' '54']
unique values in "white_blood_cell_count":
 ['7800' '6000' '7500' '6700' '7300' nan '6900' '9600' '12100' '4500'
 '12200' '11000' '3800' '11400' '5300' '9200' '6200' '8300' '8400' '10300'
 '9800' '9100' '7900' '6400' '8600' '18900' '21600' '4300' '8500' '11300'
 '7200' '7700' '14600' '6300' '\t6200' '7100' '11800' '9400' '5500' '5800'
 '13200' '12500' '5600' '7000' '11900' '10400' '10700' '12700' '6800'
 '6500' '13600' '10200' '9000' '14900' '8200' '15200' '5000' '16300'
 '12400' '\t8400' '10500' '4200' '4700' '10900' '8100' '9500' '2200'
 '12800' '11200' '19100' '\t?' '12300' '16700' '2600' '26400' '8800'
 '7400' '4900' '8000' '12000' '15700' '4100' '5700' '11500' '5400' '10800'
 '9900' '5200' '5900' '9300' '9700' '5100' '6600']
unique values in "red_blood_cell_count":
 ['5.2' nan '3.9' '4.6' '4.4' '5' '4.0' '3.7' '3.8' '3.4' '2.6' '2.8' '4.3'
 '3.2' '3.6' '4' '4.1' '4.9' '2.5' '4.2' '4.5' '3.1' '4.7' '3.5' '6.0'
 '5.0' '2.1' '5.6' '2.3' '2.9' '2.7' '8.0' '3.3' '3.0' '3' '2.4' '4.8'
 '\t?' '5.4' '6.1' '6.2' '6.3' '5.1' '5.8' '5.5' '5.3' '6.4' '5.7' '5.9'
 '6.5']
unique values in "hypertension":
 ['yes' 'no' nan]
unique values in "diabetes_mellitus":
 ['yes' 'no' ' yes' '\tno' '\tyes' nan]
unique values in "coronary_artery_disease":
 ['no' 'yes' '\tno' nan]
unique values in "appetite":
 ['good' 'poor' nan]
unique values in "peda_edema":
 ['no' 'yes' nan]
unique values in "aanemia":
 ['no' 'yes' nan]
unique values in "class":
 ['ckd' 'ckd\t' 'notckd']
```

2) Mengganti beberapa nilai yang salah

```
# replace some incorrect values
df['diabetes_mellitus'].replace(to_replace = {'\tno':'no','\tyes':'yes',' yes':'yes'},inplace=True)

df['coronary_artery_disease'] = df['coronary_artery_disease'].replace(to_replace = '\tno', value='no')

df['class'] = df['class'].replace(to_replace = {'ckd\t': 'ckd', 'notckd': 'not ckd'})
```

3) Mengonversi kolom yang diperlukan menjadi tipe numerik

```
# converting necessary columns to numerical type
df['packed_cell_volume'] = pd.to_numeric(df['packed_cell_volume'], errors='coerce')
df['white_blood_cell_count'] = pd.to_numeric(df['white_blood_cell_count'], errors='coerce')
df['red_blood_cell_count'] = pd.to_numeric(df['red_blood_cell_count'], errors='coerce')
```

```
df['class'] = df['class'].map({'not ckd': 0, 'ckd': 1})
df['class'] = pd.to_numeric(df['class'], errors='coerce')
```

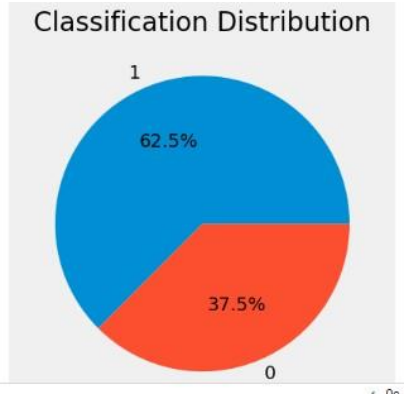
4) Mengekstraksi kolom kategorikal dan numerik

```
# Extracting categorical and numerical columns
cat_cols = [col for col in df.columns if df[col].dtype == 'object']
num_cols = [col for col in df.columns if df[col].dtype != 'object']
```

5) Distribusi data

```
# Checking target class
plt.pie(df['class'].value_counts(), labels=df['class'].value_counts().index, autopct='%1.1f%%')
plt.title('Classification Distribution')
plt.show()
```

Hasil yang diperoleh 62,5% masuk kelas 1 atau ckd sebanyak 250, 37,5% masuk kedalam kelas 0 atau not ckd sebanyak 150, Secara detail distribusi data *classification* seteleh data *preparation* dapat dilihat pada gambar 4.2.



Gambar 4 2 Distribusi Data *Classification* Seteleh Data *Preparation*

6) Memperbaiki nilai yang hilang, dimulai dengan memeriksa nilai yang kosong dengan mengetikan perintah berikut:

```
# checking for null values
df.isna().sum().sort_values(ascending = False)
```

Hasil yang diperoleh masih ada beberapa nilai yang belum terisi diantaranya red_blood_cells = 152, red_blood_cell_count = 131, white_blood_cell_count =106, potassium 88, sodium= 87, packed_cell_volume= 71, pus_cell= 65, haemoglobin= 52, sugar= 49, specific_gravity= 47, albumin= 46, blood_glucose_random= 44, blood_urea= 19,

serum_creatinine= 17, blood_pressure= 12, age= 9, bacteria = 4, pus_cell_clumps= 4, hypertension= 2, diabetes_mellitus= 2, coronary_artery_disease= 2, appetite= 1, peda_edema= 1, aanemia= 1 dapat dilihat pada gambar dibawah.

```
red_blood_cells            152
red_blood_cell_count       131
white_blood_cell_count     106
potassium                   88
sodium                      87
packed_cell_volume          71
pus_cell                    65
haemoglobin                 52
sugar                       49
specific_gravity            47
albumin                     46
blood_glucose_random        44
blood_urea                  19
serum_creatinine            17
blood_pressure              12
age                          9
bacteria                     4
pus_cell_clumps              4
hypertension                 2
diabetes_mellitus            2
coronary_artery_disease      2
appetite                     1
peda_edema                   1
aanemia                      1
class                        0
dtype: int64
```

7) Melihat nilai data numerik yang hilang

```
df[num_cols].isnull().sum().sort_values(ascending = False)
```

Hasil yang diperoleh adalah red_blood_cell_count=131, white_blood_cell_count= 106, potassium= 88, sodium= 87, packed_cell_volume= 71, haemoglobin= 52, sugar= 49, specific_gravity 47 albumin= 46, blood_glucose_random= 44, blood_urea= 19, serum_creatinine 17, blood_pressure=12, age= 9 dapat dilihat pada gambar dibawah :

```
red_blood_cell_count      131
white_blood_cell_count    106
potassium                  88
sodium                     87
packed_cell_volume         71
haemoglobin                52
sugar                      49
specific_gravity           47
albumin                    46
blood_glucose_random       44
blood_urea                 19
serum_creatinine           17
blood_pressure             12
age                         9
class                       0
dtype: int64
```

8) Melihat nilai kategorikal yang hilang

```
df[cat_cols].isnull().sum().sort_values(ascending = False)
```

Hasil yang diperoleh adalah red_blood_cells= 152, pus_cell= 65, pus_cell_clumps= 4, bacteria= 4, hypertension= 2, diabetes_mellitus= 2, coronary_artery_disease= 2, appetite= 1, peda_edema= 1, aanemia = 1, dapat dilihat pada gambar di bawah.

```
red_blood_cells           152
pus_cell                   65
pus_cell_clumps             4
bacteria                    4
hypertension                2
diabetes_mellitus           2
coronary_artery_disease     2
appetite                    1
peda_edema                  1
aanemia                     1
dtype: int64
```

9) Mengisi nilai yang kosong, kita akan menggunakan dua metode, pengambilan sampel acak untuk nilai kosong yang lebih tinggi dan pengambilan sampel mean/mode untuk nilai kosong yang lebih rendah.

```
# filling null values, we will use two methods, random sampling for higher null
def random_value_imputation(feature):
    random_sample = df[feature].dropna().sample(df[feature].isna().sum())
    random_sample.index = df[df[feature].isnull()].index
    df.loc[df[feature].isnull(), feature] = random_sample

def impute_mode(feature):
    mode = df[feature].mode()[0]
    df[feature] = df[feature].fillna(mode)
```

10) Mengisi nilai yang kosong num_cols menggunakan metode pengambilan sampel acak

```
# filling num_cols null values using random sampling method
for col in num_cols:
    random_value_imputation(col)

df[num_cols].isnull().sum()
```

```
age                     0
blood_pressure          0
specific_gravity        0
albumin                 0
sugar                   0
blood_glucose_random    0
blood_urea              0
serum_creatinine        0
sodium                  0
potassium               0
haemoglobin             0
packed_cell_volume      0
white_blood_cell_count  0
red_blood_cell_count    0
class                   0
dtype: int64
```

11) mengisi "red_blood_cell" dan "pus_cell" menggunakan metode pengambilan sampel acak dan sisa cat_cols menggunakan mode imputasi

```
# filling "red_blood_cells" and "pus_cell" using random sampling
random_value_imputation('red_blood_cells')
random_value_imputation('pus_cell')

for col in cat_cols:
    impute_mode(col)

df[cat_cols].isnull().sum()
```

Hasil yang diperoleh adalah red_blood_cells= 0, pus_cell= 0, pus_cell_clumps= 0, bacteria= 0, hypertension= 0, diabetes_mellitus= 0, coronary_artery_disease= 0, appetite= 0, peda_edema= 0, aanemia = 0, dapat dilihat pada gambar dibawah.

```
red_blood_cells           0
pus_cell                  0
pus_cell_clumps           0
bacteria                  0
hypertension              0
diabetes_mellitus         0
coronary_artery_disease   0
appetite                  0
peda_edema                0
aanemia                   0
dtype: int64
```

4.1.3 Seleksi Fitur

Seleksi Fitur adalah salah satu teknik yang sangat penting serta sering dipakai pada pre-processing. Pada teknik ini dilakukan pengurangan jumlah fitur yang terlibat agar dapat menentukan nilai kelas target dengan cara mengurangi fitur yang tidak sesuai dan data yang berlebihan. pemilihan fitur adalah untuk meningkatkan kemampuan pembelajaran mesin dengan meningkatkan kekuatan prediktif dan mengurangi biaya waktu. Disini saya menggunakan kategori pemilihan fitur berbasis statistik. Pemilihan fitur berbasis statistic memberi kita metode yang relatif cepat dan mudah dalam menafsirkan data kuantitatif dan kualitatif. Untuk memilih fitur saya menggunakan korelasi pearson (*Pearson correlation*) karena memiliki kelebihan utama yaitu mudah dihitung dan diinterpretasikan,dengan perintah sebagai berikut :

```
# Melihat korelasi atribut terhadap kelas
# just correlations between every feature and the response
df.corr()['class']
```

Hasil korelasi atribut terhadap kelas :

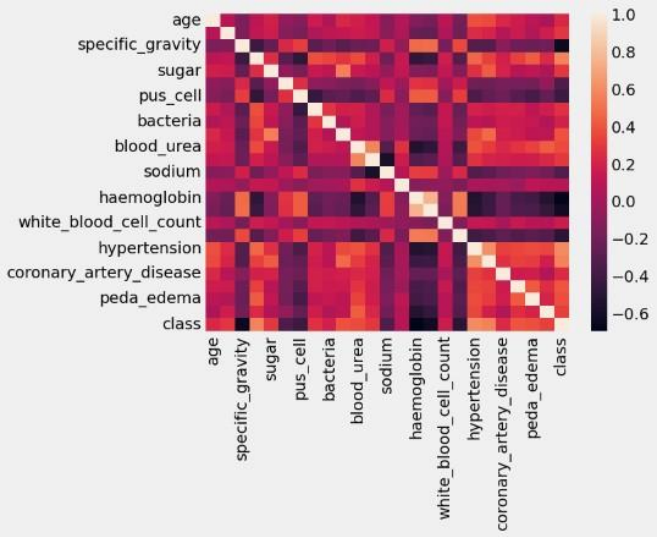
```
age                      0.227107
blood_pressure           0.285392
specific_gravity        -0.635447
albumin                  0.565181
sugar                    0.293926
red_blood_cells         -0.348828
pus_cell                -0.420356
pus_cell_clumps          0.265313
bacteria                 0.186871
blood_glucose_random     0.384831
blood_urea               0.359009
serum_creatinine         0.297553
sodium                  -0.332109
potassium                0.073920
haemoglobin             -0.694497
packed_cell_volume      -0.616607
white_blood_cell_count   0.162769
red_blood_cell_count    -0.499448
hypertension             0.590438
diabetes_mellitus        0.559060
coronary_artery_disease  0.236088
appetite                 0.393341
peda_edema               0.375154
aanemia                  0.325396
class                    1.000000
Name: class, dtype: float64
```

Koefisien korelasi Pearson mengukur hubungan linier antar kolom. Nilai koefisien bervariasi antara -1 dan +1, dimana 0 berarti tidak ada korelasi di antara keduanya. Korelasi yang mendekati -1 atau +1 menunjukkan hubungan linier yang sangat kuat.

Dengan fasilitas fungsi heatmap secara otomatis memilih fitur yang paling berkorelasi untuk ditampilkan kepada kita, dengan memberi perintah sebagai berikut :

```
# Visualisasi Korelasi
# using seaborn to generate heatmaps
import seaborn as sns
import matplotlib.style as style
# Use a clean stylizatino for our charts and graphs
style.use('fivethirtyeight')
```

Hasil yang diperoleh visualisasi korelasi dapat dilihat pada gambar 4.3.



Gambar 4 3 Visualisasi Korelasi

Selanjutnya kita dapat melihat dan mengurutkan korelasi dari terbesar ke terkecil dengan perintah sebagai berikut :

```
# Melihat Daftar Korelasi
target_corr = df.corr()['class'].abs().sort_values(ascending=False)[1:]
print(target_corr)
```

Hasil pengurutan korelasi yang diperoleh :

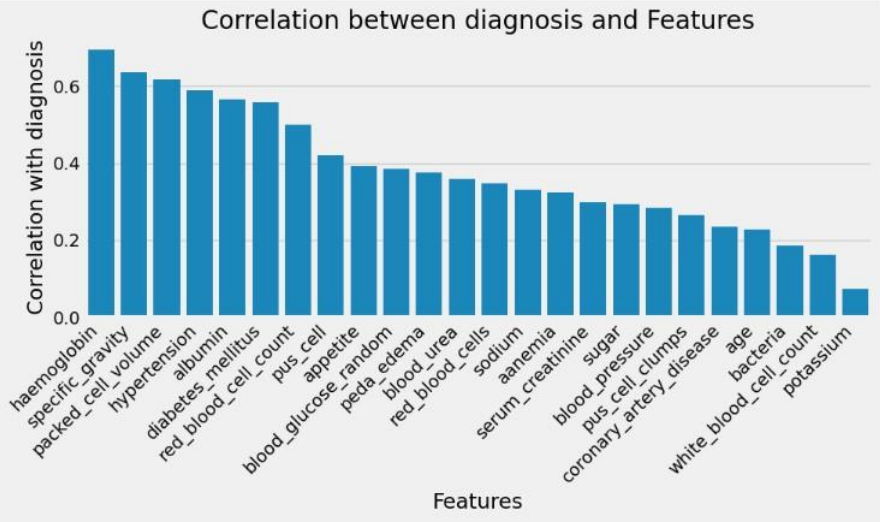
```
haemoglobin                0.694497
specific_gravity           0.635447
packed_cell_volume         0.616607
hypertension               0.590438
albumin                    0.565181
diabetes_mellitus          0.559060
red_blood_cell_count       0.499448
pus_cell                   0.420356
appetite                   0.393341
blood_glucose_random       0.384831
peda_edema                 0.375154
blood_urea                 0.359009
red_blood_cells            0.348828
sodium                     0.332109
aanemia                    0.325396
serum_creatinine           0.297553
sugar                      0.293926
blood_pressure             0.285392
pus_cell_clumps            0.265313
coronary_artery_disease    0.236088
age                        0.227107
bacteria                   0.186871
white_blood_cell_count     0.162769
potassium                  0.073920
Name: class, dtype: float64
```

Koefisien korelasi Pearson mengukur hubungan linier antar kolom. Nilai koefisien koerelasi tertinggi adalah haemoglobin, sebesar 0.694497 dan terkecil nilai korelasinya adalah potassium sebesar 0.073920

Kita juga dapat melihat korelasi dalam bentuk grafik batang dari yang tertinggi hingga yang terendah dengan mengetikkan perintah sebagai berikut :

```
# Membuat diagram batang untuk memvisualisasikan korelasinya
plt.figure(figsize=(10, 6))
sns.barplot(x=target_corr.index, y=target_corr.values)
plt.xticks(rotation=45, ha='right')
plt.xlabel('Features')
plt.ylabel('Correlation with diagnosis')
plt.title('Correlation between diagnosis and Features')
plt.tight_layout()
plt.show()
```

Hasil yang diperoleh korelasi secara urut dapat dilihat pada gambar 4.4. :



Gambar 4 4 Korelasi Secara Urut

Selanjutnya memisahkan nilai korelasi diatas 0.5 dengan perintah sebagai berikut :

```
# Memisahkan fiture yang memiliki korelasi > 0.5
# filter only correlations stronger than .5 in either direction (positive or negative)
df.corr()['class'].abs() > .5
```

Hasil yang diperoleh :

```
age                      False
blood_pressure           False
specific_gravity          True
albumin                   True
sugar                    False
red_blood_cells          False
pus_cell                 False
pus_cell_clumps          False
bacteria                 False
blood_glucose_random     False
blood_urea               False
serum_creatinine         False
sodium                   False
potassium                False
haemoglobin               True
packed_cell_volume        True
white_blood_cell_count   False
red_blood_cell_count     False
hypertension              True
diabetes_mellitus         True
coronary_artery_disease  False
appetite                 False
peda_edema               False
aanemia                  False
class                     True
Name: class, dtype: bool
```

Menampilkan fitur dengan korelasi > 0.5

```
highly_correlated_features
```

Hasil yang diperoleh sebagai berikut:

```
Index(['specific_gravity', 'albumin', 'haemoglobin', 'packed_cell_volume',
       'hypertension', 'diabetes_mellitus', 'class'],
      dtype='object')
```

Diperoleh tujuh (7) fitur yang memiliki korelasi > 0.5 yaitu *specific_gravity, albumin, haemoglobin, packed_cell_volume, hypertension, diabetes_mellitus* dan *class.*

Langkah selanjutnya menghapus variable respon dengan perintah sebagai berikut :

```
# drop the response variable
highly_correlated_features = highly_correlated_features.drop('class')
```

Menampilkan fitur dengan korelasi > 0.5

```
highly_correlated_features
```

Hasil yang diperoleh enam (6) fitur yang memiliki korelasi > 0.5 dapat dilihat pada tabel 4.1 sebagai berikut:

Tabel 4.1 fitur yang memiliki korelasi > 0.5

No	Fitur
1	*specific_gravity*
2	*Albumin*
3	*Haemoglobin*
4	*packed_cell_volume*
5	*hypertension*
6	*diabetes_mellitus*

4.1.4 Split Data

Split data atau pemisahan data merupakan metode membagi data menjadi dua bagian. disini, data splitting memisahkan dua bagian, bagian pertama digunakan untuk mengevaluasi atau uji sebesar 20% data dan data lainnya digunakan untuk melatih model sebesar 80 %, dapat kita ketikan perintah sebagai berikut :

```
# X = feature matrix
x = df.drop("class", axis=1)     # everything except 'class' column

# Y = lables
y = df['class']      # only 'class' column

# spliting data to train and test
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
```

Split data pada klasifikasi dengan seleksi fitur, dimana data latih sebesar 80%, dan data uji sebesar 20%, dapat diketikan perintah sebagai berikut :

```
# X = feature matrix
X = df.drop("class", axis=1)     # everything except 'class' column
# Y = lables
Y = df['class']      # only 'class' column
# only include the six highly correlated features
X_subsetted = X[highly_correlated_features]


# spliting data to train and test
X_latih, X_uji, y_latih, y_uji = train_test_split(X_subsetted, Y, test_size=0.2, random_state=42)
```

4.1.5 Klasifikasi C4.5

Pada tahap ini dilakukan pengujian data penyakit ginjal kronis menggunakan algoritma C4.5, dengan kita ketikan perintah sebagai berikut :

```
# Decision Tree
dtc = DecisionTreeClassifier()
dtc.fit(x_train, y_train)
```

4.1.6 Klasifikasi *Random Forest*

Pada tahap ini dilakukan pengujian data penyakit ginjal kronis menggunakan

algoritma *Random Forest* dengan kita ketikan perintah sebagai berikut :

```
# Random Forest
rfc = RandomForestClassifier()
rfc.fit(x_train, y_train)
```

4.1.7 Klasifikasi C4.5 dengan Seleksi Fitur

Pada tahap ini dilakukan pengujian data penyakit ginjal kronis menggunakan algoritma C4.5 dengan seleksi fitur dengan mengetikan perintah sebagai berikut :

```
# Decision Tree
dtc_fs = DecisionTreeClassifier()
dtc_fs.fit(X_latih, y_latih)
```

4.1.8 Klasifikasi *Random forest* dengan Seleksi Fitur

Pada tahap ini dilakukan pengujian data penyakit ginjal kronis menggunakan algoritma *Random Forest* dengan seleksi fitur dengan seleksi fitur dengan mengetikan perintah sebagai berikut :

```
# Random Forest
rfc_fs = RandomForestClassifier()
rfc_fs.fit(X_latih, y_latih)
```

4.2 Hasil Penelitian

Pada penelitian yang telah dilakukan menggunakan algoritme C.4.5 dan *Random Forest* serta algoritme C.4.5 dengan seleksi fitur dan*Random Forest* dengan seleksi fitur menghasilkan nilai Akurasi, Confusion Metrik, Presisi, Recal, F1 score serta Kurva ROC.

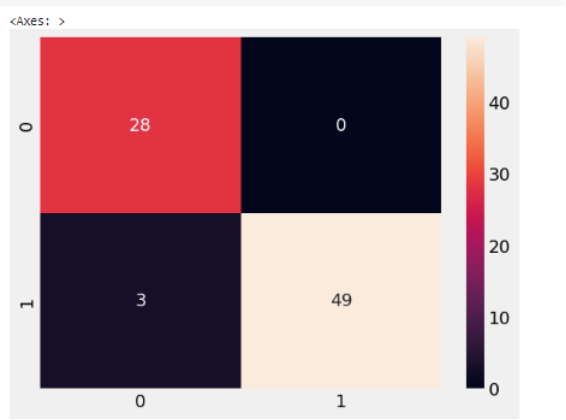
4.2.1 Algoritme C.4.5

4.2.1.1 Confusion Matrix

Pengujian pertama adalah menguji data dengan menggunakan algoritme C.4.5, memperoleh nilai *Confusion Matrix* dengan mengetikkan perintah berikut :

```
cf_matrix=confusion_matrix(y_test, y_preda)
sns.heatmap(cf_matrix, annot=True)
```

Hasil yang diperoleh *confusion matrix* algoritme C.4.5 dapat dilihat pada gambar 4.5.



Gambar 4 5 *Confusion Matrix* Algoritme C.4.5

4.2.1.2 Akurasi

Pada pengujian ini memperoleh nilai akurasi sebesar 96,25

```
# from sklearn.tree import DecisionTreeClassifier
y_preda=dtc.predict(x_test)
print(accuracy_score(y_test, y_preda))
```

```
0.9625
```

Pada pengujian ini untuk memperoleh nilai $acuracy$ menggunakan persamaan 5 sebagai berikut :

$$Accuracy = \frac{28+49}{28+3+49+0}$$

$$Accuracy = 96{,}25$$

4.2.1.3 Precission

Pada pengujian ini nilai $precission$ dapat diperoleh dengan persamaan 6 sebagai berikut :

$$Precission = \frac{28}{28+3}$$

$$Precission = 90{,}32$$

4.2.1.4 Recall

Pada pengujian ini nilai $recall$ dapat diperoleh dengan persamaan 7 sebagai berikut :

$$Recall = \frac{28}{28+0}$$

$$Recall = 100$$

4.2.1.5 F1 Score

Pada pengujian ini nilai $f1\ score$ dapat diperoleh dengan persamaan 8 sebagai berikut :

$$F1\ Score = 2 * \frac{1*90{,}32}{1+90{,}32}$$

$$F1\ Score = 94{,}92$$

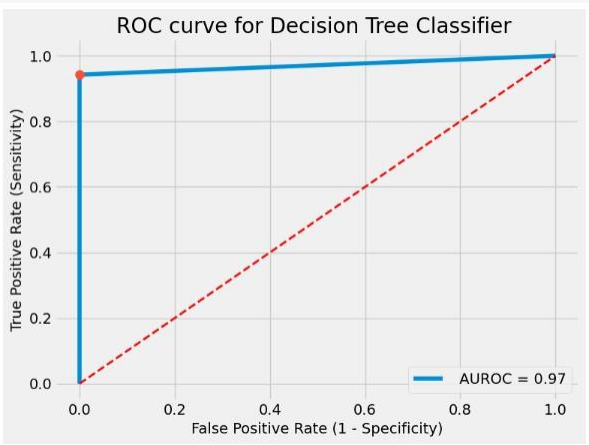
4.2.1.6 Kurva ROC (*Receiver Operating Characteristic*)

Receiver Operating Characteristic (ROC) merupakan hasil dari pengukuran klasifikasi dalam bentuk 2 dimensi dimana garis horisontal menggambarkan nilai false positive dan garis vertikal sebagai true positive, dapat ditampilakn dengan

perintah sebagai berikut :

```
fig, (ax2) = plt.subplots(figsize = (8,6))
        #roc-curve
fpr, tpr, thresholds_roc = roc_curve(y_test,y_preda)
roc_auc = auc(fpr,tpr)
ax2.plot(fpr,tpr, label = " AUROC = {:0.2f}".format(roc_auc))
ax2.plot([0,1], [0,1], 'r', linestyle = "--", lw = 2)
ax2.set_xlabel("False Positive Rate", fontsize = 14)
ax2.set_ylabel("True Positive Rate", fontsize = 14)
ax2.set_title("ROC Curve", fontsize = 18)
ax2.legend(loc = 'best')
plt.title('ROC curve for Decision Tree Classifier ')
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
        #find default threshold
close_default = np.argmin(np.abs(thresholds_roc - 0.5))
ax2.plot(fpr[close_default], tpr[close_default], 'o', markersize = 8)
plt.tight_layout()
```

Hasil yang diperoleh r*eceiver operating characteristic* (ROC) algoritme C.4.5 dapat dilihat pada gambar 4.6.



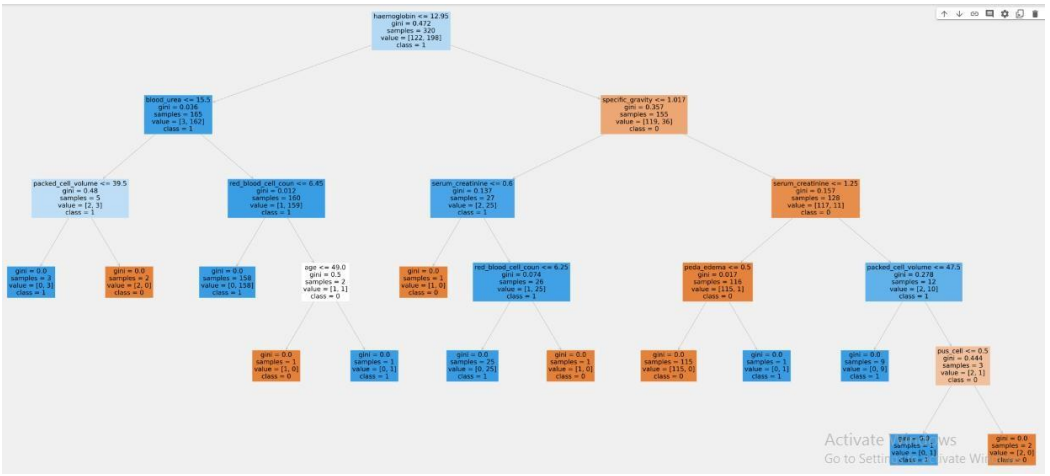
Gambar 4 6 *Receiver Operating Characteristic* (ROC) Algoritme C.4.5

4.2.1.7 Pohon Keputusan

Selanjutnya untuk melihat pohon keputusan pada algoritme C.4.5 dapat diketikan perintah sebagai berikut :

```
from matplotlib import pyplot as plt
fig = plt.figure(figsize=(100,50))
_ = tree.plot_tree(dtc,
                   feature_names=['age','blood_pressure','specific_gravity','albumin','sugar','red_blood_cells',
                                  'pus_cell','pus_cell_clumps','bacteria','blood_glucose_random','blood_urea',
                                  'serum_creatinine','sodium','potassium','haemoglobin','packed_cell_volume',
                                  'white_blood_cell_count','red_blood_cell_coun','hypertension',
                                  'diabetes_mellitus','coronary_artery_disease','appetite','peda_edema','aanemia'],
                   class_names=['0','1'],
                   filled=True)
```

Hasil pohon keputusan algoritme C.4.5 dapat dilihat pada gambar 4.7.



Gambar 4 7 Pohon Keputusan Algoritme C.4.5

4.2.1.8 Rule

Selain pohon keputusan, pengujian model C4.5 dalam Google Colaboratory tersebut juga secara otomatis menghasilkan aturan-aturan atau rule dengan mengetikan perintah sebagai berikut :

```
from sklearn import tree
text_representation = tree.export_text(dtc)
print(text_representation)
```

Hasil rule yang diperoleh sebagai berikut :

```
|--- feature_14 <= 12.95
|  |--- feature_10 <= 15.50
|  |   |--- feature_15 <= 39.50
|  |   |   |--- class: 1
|  |   |--- feature_15 >  39.50
|  |   |   |--- class: 0
|  |--- feature_10 >  15.50
|  |   |--- feature_17 <= 6.45
|  |   |   |--- class: 1
|  |   |--- feature_17 >  6.45
|  |   |   |--- feature_0 <= 49.00
|  |   |   |   |--- class: 0
|  |   |   |--- feature_0 >  49.00
|  |   |   |   |--- class: 1
|--- feature_14 >  12.95
|  |--- feature_2 <= 1.02
|  |   |--- feature_11 <= 0.60
|  |   |   |--- class: 0
|  |   |--- feature_11 >  0.60
|  |   |   |--- feature_17 <= 6.25
|  |   |   |   |--- class: 1
|  |   |   |--- feature_17 >  6.25
|  |   |   |   |--- class: 0
```

```
| |--- feature_2 >  1.02
| |   |--- feature_11 <= 1.25
| |   |   |--- feature_22 <= 0.50
| |   |   |   |--- class: 0
| |   |   |--- feature_22 >  0.50
| |   |   |   |--- class: 1
| |   |--- feature_11 >  1.25
| |   |   |--- feature_15 <= 47.50
| |   |   |   |--- class: 1
| |   |   |--- feature_15 >  47.50
| |   |   |   |--- feature_6 <= 0.50
| |   |   |   |   |--- class: 1
| |   |   |   |--- feature_6 >  0.50
| |   |   |   |   |--- class: 0
```

Berdasarkan aturan-aturan atau rule yang dihasilkan, terdapat 13 rule yang terbentuk dari pohon keputusan algoritma klasifikasi C4.5, dengan jumlah class 1 sebanyak 7 rule dan 6 rule untuk class 0 pada prediksi penyakit ginjal kronis atau *Chronic Kidney Disease* (CKD).

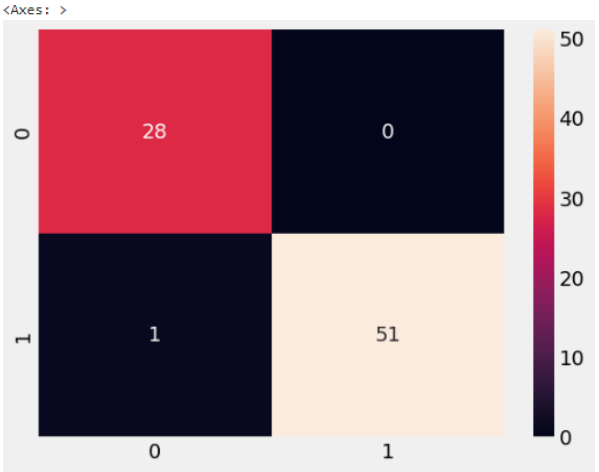
4.2.2 Algoritme *Random Forest*

4.2.2.1 Confusion Matrix

Pengujian pertama adalah menguji data dengan menggunakan algoritme ***Random Forest***, memperoleh nilai *Confusion Matrix* dengan mengetikkan perintah berikut

```
cf_matrix=confusion_matrix(y_test, y_predb)
sns.heatmap(cf_matrix, annot=True)
```

Hasil yang diperoleh *confusion matrix Random Forest* dapat dilihat pada gambar 4.8.



Gambar 4 8 *Confusion Matrix Random Forest*

4.2.2.2 Accuracy

Pada pengujian ini memperoleh nilai akurasi sebesar 98,75

```
from sklearn.ensemble import RandomForestClassifier
y_predb=rfc.predict(x_test)
print(accuracy_score(y_test, y_predb))
```

```
0.9875
```

Pada pengujian ini untuk memperoleh nilai $acuracy$ menggunakan persamaan 5 sebagai berikut :

$$Accuracy = \frac{28+51}{28+1+51+0}$$

$$Accuracy = 98{,}75$$

4.2.2.3 Precission

Pada pengujian ini nilai $precission$ dapat diperoleh dengan persamaan 6 sebagai berikut :

$$Precission = \frac{28}{28+1}$$

$$Precission = 96{,}55$$

4.2.2.4 Recall

Pada pengujian ini nilai $recall$ dapat diperoleh dengan persamaan 7 sebagai berikut :

$$Recall = \frac{28}{28+0}$$

$$Recall = 100$$

4.2.2.5 F1 Score

Pada pengujian ini nilai $f1\ score$ dapat diperoleh dengan persamaan 8 sebagai berikut :

$$F1\ Score\ = 2 * \frac{1*96,55}{1+96,55}$$

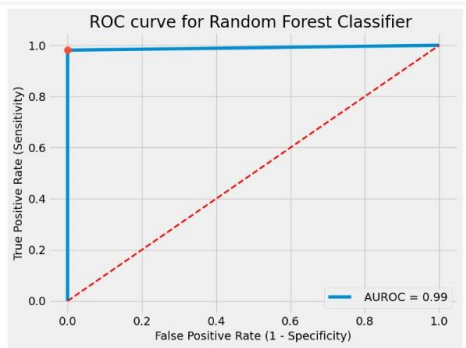
$$F1\ Score = 98,25$$

4.2.2.6 Kurva ROC

Pada pengujian menggunakan Kurva *Receiver Operating Characteristic* (ROC) adalah dengan mengetikan perintah sebagai berikut :

```
fig, (ax2) = plt.subplots(figsize = (8,6))
        #roc-curve
fpr, tpr, thresholds_roc = roc_curve(y_test,y_predb)
roc_auc = auc(fpr,tpr)
ax2.plot(fpr,tpr, label = " AUROC = {:0.2f}".format(roc_auc))
ax2.plot([0,1], [0,1], 'r', linestyle = "--", lw = 2)
ax2.set_xlabel("False Positive Rate", fontsize = 14)
ax2.set_ylabel("True Positive Rate", fontsize = 14)
ax2.set_title("ROC Curve", fontsize = 18)
ax2.legend(loc = 'best')
plt.title('ROC curve for Random Forest Classifier ')
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
        #find default threshold
close_default = np.argmin(np.abs(thresholds_roc - 0.5))
ax2.plot(fpr[close_default], tpr[close_default], 'o', markersize = 8)
plt.tight_layout()
```

Hasil yang diperoleh *receiver operating characteristic* (ROC) Algoritme *Random Forest* dapat dilihat pada gambar 4.9.



Gambar 4 9 *Receiver Operating Characteristic* (ROC) Algoritme *Random Forest*

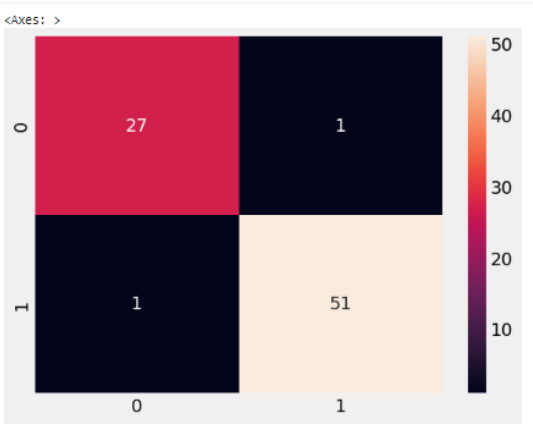
4.2.3 Algoritme C.4.5 dengan Seleksi Fitur

4.2.3.1 Confusion Matrix

Pada pengujian ini kita menguji data dengan menggunakan algoritme ***C4.5*** dengan seleksi fitur, untuk nilai *Confusion Matrix* dengan mengetikkan perintah berikut :

```
cf_matrix=confusion_matrix(y_uji, y_pred01)
sns.heatmap(cf_matrix, annot=True)
```

Hasil yang diperoleh *confusion matrix* Algortime C.4.5 dengan Seleksi Fitur dapat dilihat pada gambar 4.10.



Gambar 4 10 *Confusion Matrix* Algortime C.4.5 dengan Seleksi Fitur

4.2.3.2 Accuracy

Pada pengujian ini, kita mencari nilai akurasi dan diperoleh nilai sebesar 97,50, dengan cara mengetikkan perintah sebagai berikut:

```
from sklearn.tree import DecisionTreeClassifier
y_pred01=dtc_fs.predict(X_uji)
print(accuracy_score(y_uji, y_pred01))
```

```
0.975
```

Pada pengujian ini untuk memperoleh nilai $acuracy$ menggunakan persamaan 5 sebagai berikut :

$$Accuracy = \frac{27+51}{27+1+51+1}$$

$Accuracy = 97{,}50$

4.2.3.3 Precission

Pada pengujian ini nilai $precission$ dapat diperoleh dengan persamaan 6 sebagai berikut :

$Precission = \frac{27}{27+1}$

$Precission = 96{,}43$

4.2.3.4 Recall

Pada pengujian ini nilai $recall$ dapat diperoleh dengan persamaan 7 sebagai berikut :

$Recall = \frac{27}{27+1}$

$Recall = 96{,}43$

4.2.3.5 F1 Score

Pada pengujian ini nilai $f1\ score$ dapat diperoleh dengan persamaan 8 sebagai berikut :

$F1\ Score = 2 * \frac{96{,}43*96{,}43}{96{,}43+96{,}43}$

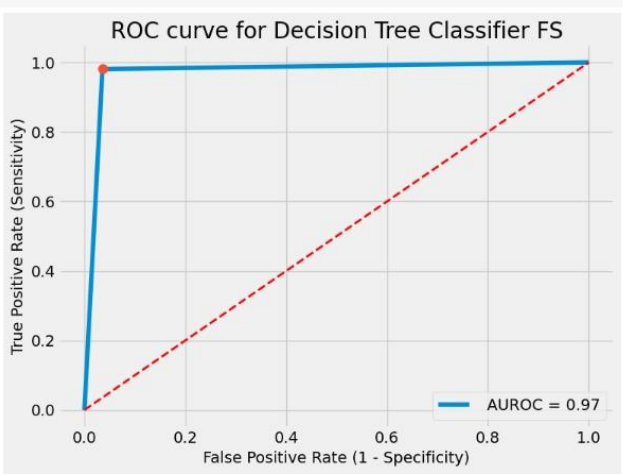
$F1\ Score = 96{,}43$

4.2.3.6 Kurva ROC

Pada pengujian berikutnya untuk memperoleh nilai Kurva *Receiver Operating Characteristic* (ROC) adalah dengan mengetikan perintah sebagai berikut :

```
fig, (ax2) = plt.subplots(figsize = (8,6))
        #roc-curve
fpr, tpr, thresholds_roc = roc_curve(y_uji,y_pred01)
roc_auc = auc(fpr,tpr)
ax2.plot(fpr,tpr, label = " AUROC = {:0.2f}".format(roc_auc))
ax2.plot([0,1], [0,1], 'r', linestyle = "--", lw = 2)
ax2.set_xlabel("False Positive Rate", fontsize = 14)
ax2.set_ylabel("True Positive Rate", fontsize = 14)
ax2.set_title("ROC Curve", fontsize = 18)
ax2.legend(loc = 'best')
plt.title('ROC curve for Decision Tree Classifier FS')
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
        #find default threshold
close_default = np.argmin(np.abs(thresholds_roc - 0.5))
ax2.plot(fpr[close_default], tpr[close_default], 'o', markersize = 8)
plt.tight_layout()
```

Hasil yang diperoleh r*eceiver operating characteristic* (ROC) Algoritme C.4.5 dengan Seleksi Fitur dapat dilihat pada gambar 4.11.

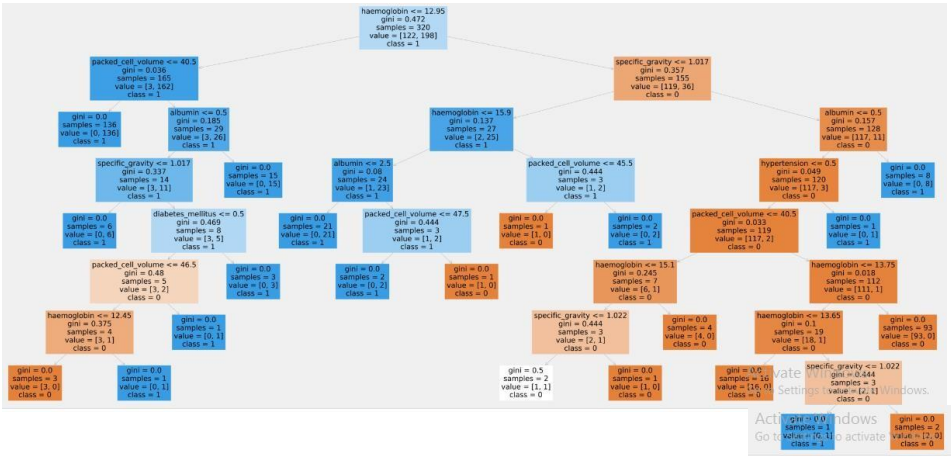


Gambar 4 11 *Receiver Operating Characteristic* (ROC) Algoritme C.4.5 dengan Seleksi Fitur

4.2.3.7 Pohon Keputusan

```
from matplotlib import pyplot as plt
fig = plt.figure(figsize=(100,50))
_ = tree.plot_tree(dtc_fs,
                   feature_names=['specific_gravity', 'albumin', 'haemoglobin', 'packed_cell_volume',
       'hypertension', 'diabetes_mellitus'],
                   class_names=['0','1'],
                   filled=True)
```

Hasil pohon keputusan algoritme C.4.5 dengan seleksi fitur dapat dilihat pada gambar 4.12.



Gambar 4 12 Pohon Keputusan Algoritme C.4.5 dengan Seleksi Fitur

4.2.3.8 Rule

Pengujian untuk menghasilkan rule atau aturan-aturan adalah dengan mengetikan perintah sebagai berikut :

```
from sklearn import tree
text_representation = tree.export_text(dtc_fs)
print(text_representation)
```

Hasil yang diperoleh adalah sebagai berikut :

```
|--- feature_2 <= 12.95
| |--- feature_3 <= 40.50
| |   |--- class: 1
| |--- feature_3 >  40.50
| |   |--- feature_1 <= 0.50
| |   |   |--- feature_0 <= 1.02
| |   |   |   |--- class: 1
| |   |   |--- feature_0 >  1.02
| |   |   |   |--- feature_5 <= 0.50
| |   |   |   |   |--- feature_3 <= 46.50
| |   |   |   |   |   |--- feature_2 <= 12.45
| |   |   |   |   |   |   |--- class: 0
| |   |   |   |   |   |--- feature_2 >  12.45
| |   |   |   |   |   |   |--- class: 1
| |   |   |   |   |--- feature_3 >  46.50
| |   |   |   |   |   |--- class: 1
| |   |   |   |--- feature_5 >  0.50
| |   |   |   |   |--- class: 1
| |   |--- feature_1 >  0.50
| |   |   |--- class: 1
|--- feature_2 >  12.95
| |--- feature_0 <= 1.02
| |   |--- feature_2 <= 15.90
| |   |   |--- feature_1 <= 2.50
| |   |   |   |--- class: 1
| |   |   |--- feature_1 >  2.50
| |   |   |   |--- feature_3 <= 47.50
```

```
| | | | | |--- class: 1
| | | | |--- feature_3 >  47.50
| | | | | |--- class: 0
| | |--- feature_2 >  15.90
| | | |--- feature_3 <= 45.50
| | | | |--- class: 0
| | | |--- feature_3 >  45.50
| | | | |--- class: 1
| |--- feature_0 >  1.02
| | |--- feature_1 <= 0.50
| | | |--- feature_4 <= 0.50
| | | | |--- feature_3 <= 40.50
| | | | | |--- feature_2 <= 15.10
| | | | | | |--- feature_0 <= 1.02
| | | | | | | |--- class: 0
| | | | | | |--- feature_0 >  1.02
| | | | | | | |--- class: 0
| | | | | |--- feature_2 >  15.10
| | | | | | |--- class: 0
| | | | |--- feature_3 >  40.50
| | | | | |--- feature_2 <= 13.75
| | | | | | |--- feature_2 <= 13.65
| | | | | | | |--- class: 0
| | | | | | |--- feature_2 >  13.65
| | | | | | | |--- feature_0 <= 1.02
| | | | | | | | |--- class: 1
| | | | | | | |--- feature_0 >  1.02
| | | | | | | | |--- class: 0
| | | | | |--- feature_2 >  13.75
| | | | | | |--- class: 0
| | | |--- feature_4 >  0.50
| | | | |--- class: 1
| | |--- feature_1 >  0.50
| | | |--- class: 1
```

Berdasarkan aturan-aturan atau rule yang dihasilkan, terdapat 21 rule yang terbentuk dari pohon keputusan algoritma klasifikasi C4.5 dengan seleksi fitur, dengan jumlah class 1 sebanyak 12 rule dan 9 rule untuk class 0 pada prediksi penyakit ginjal kronis atau *Chronic Kidney Disease* (CKD).

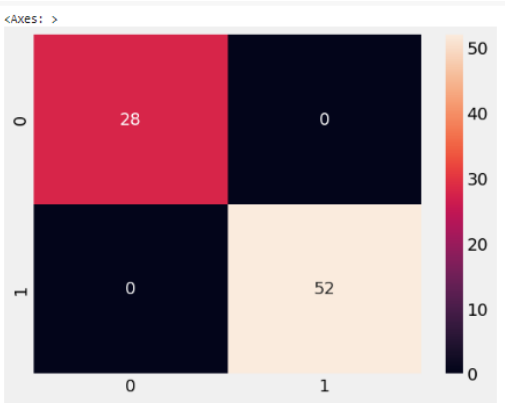
4.2.4 Algoritme *Random Forest* dengan Seleksi Fitur

4.2.4.1 Confusion Matrix

Pada pengujian algoritme ***Random Forest*** dengan seleksi fitur memperoleh nilai *Confusion Matrix* dengan mengetikkan perintah berikut :

```
cf_matrix=confusion_matrix(y_uji, y_pred02)
sns.heatmap(cf_matrix, annot=True)
```

Hasil yang diperoleh *confusion matrix* algoritme *random forest* dengan seleksi fitur dapat dilihat pada gambar 4.13.



Gambar 4 13 *Confusion Matrix* Algoritme *Random Forest* dengan Seleksi Fitur

4.2.4.2 Accuracy

Pada pengujian ini memperoleh nilai akurasi sebesar 100

```
from sklearn.ensemble import RandomForestClassifier
y_pred02=rfc_fs.predict(X_uji)
print(accuracy_score(y_uji, y_pred02))
```

```
1.0
```

Pada pengujian ini untuk memperoleh nilai $acuracy$ menggunakan persamaan 5 sebagai berikut :

$$Accuracy = \frac{28+52}{28+0+52+0}$$

$$Accuracy = 100$$

4.2.4.3 Precission

Pada pengujian ini nilai $precission$ dapat diperoleh dengan persamaan 6 sebagai berikut :

$$Precission = \frac{28}{28+0}$$

$$Precission = 100$$

4.2.4.4 *Recall*

Pada pengujian ini nilai $recall$ dapat diperoleh dengan persamaan 7 sebagai berikut :

$$Recall = \frac{28}{28+0}$$

$$Recall = 100$$

4.2.4.5 F1 *Score*

Pada pengujian ini nilai $f1\ score$ dapat diperoleh dengan persamaan 8 sebagai berikut :

$$F1\ Score = 2 * \frac{100*100}{100+100}$$

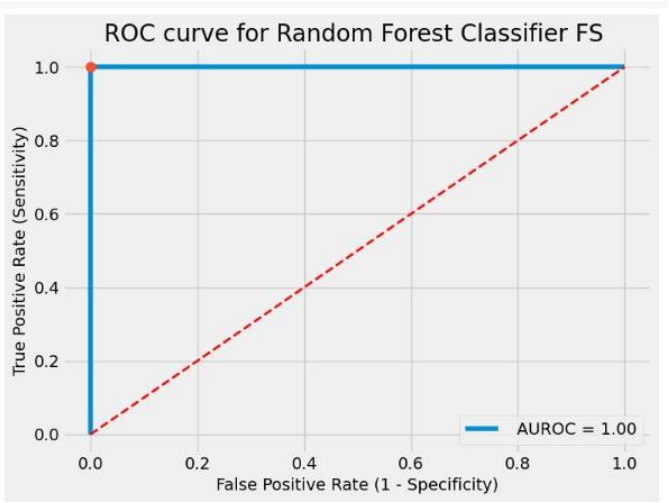
$$F1\ Score = 100$$

4.2.4.6 Kurva ROC

Pada pengujian agar menghasilkan Kurva *Receiver Operating Characteristic* (ROC) adalah dengan mengetikan perintah sebagai berikut :

```
fig, (ax2) = plt.subplots(figsize = (8,6))
        #roc-curve
fpr, tpr, thresholds_roc = roc_curve(y_uji,y_pred02)
roc_auc = auc(fpr,tpr)
ax2.plot(fpr,tpr, label = " AUROC = {:0.2f}".format(roc_auc))
ax2.plot([0,1], [0,1], 'r', linestyle = "--", lw = 2)
ax2.set_xlabel("False Positive Rate", fontsize = 14)
ax2.set_ylabel("True Positive Rate", fontsize = 14)
ax2.set_title("ROC Curve", fontsize = 18)
ax2.legend(loc = 'best')
plt.title('ROC curve for Random Forest Classifier FS ')
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
        #find default threshold
close_default = np.argmin(np.abs(thresholds_roc - 0.5))
ax2.plot(fpr[close_default], tpr[close_default], 'o', markersize = 8)
plt.tight_layout()
```

Hasil yang diperoleh *receiver operating characteristic* (ROC) algoritme *random forest* dengan seleksi fitur dapat dilihat pada gambar 4.14.



Gambar 4 14 *Receiver Operating Characteristic* (ROC) Algoritme *Random Forest* dengan Seleksi Fitur

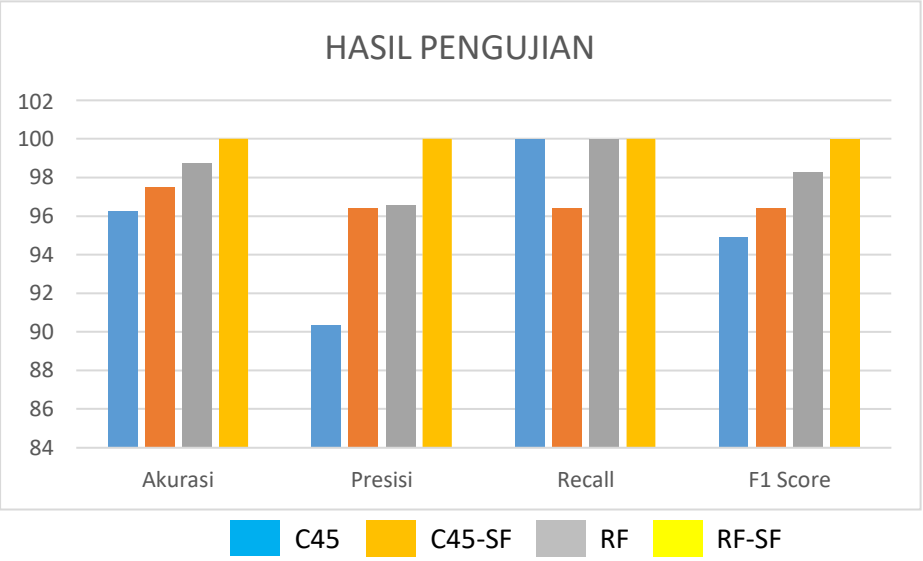
4.3 Pembahasan Hasil Penelitian

Setelah melakukan pengujian, dapat ditampilkan nilai akurasi, presisi, recal dan f1 score. Secara detail hasil pengujian dapat dilihat pada tabel 4.2.

Tabel 4 2 Hasil Pengujian

Nama Algoritme	Akurasi	Presisi	Recal	F1 Score
C45	96,25	90,32	100	94,92
C45-FS	97,50	96,43	96,43	96,43
RF	98,75	96,55	100	98,25
RF-FS	100	100	100	100

Nilai hasil pengujian dapat ditampilkan dalam bentuk grafik yang dapat dilihat pada gambar 4.15.



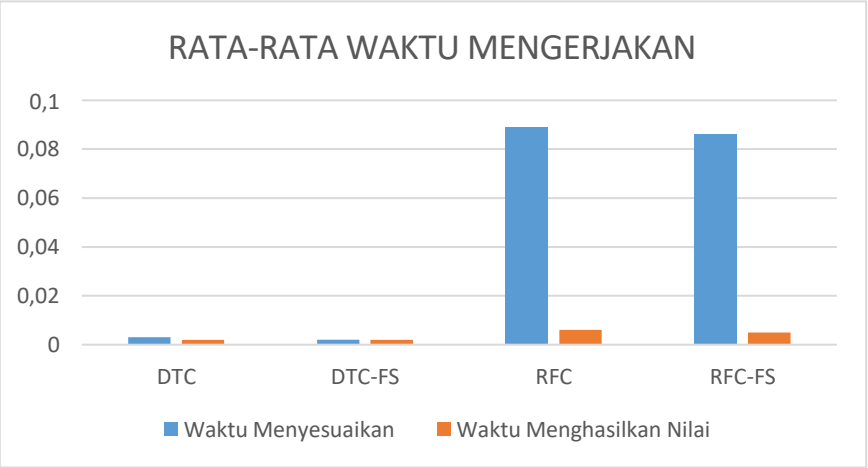
Gambar 4. 15 Grafik Perbandingan Hasil Pengujian

Selain nilai akurasi dapat pula ditampilkan nilai waktu menyesuaikan dan waktu mencetak nilai pada algoritme C.4.5 dan *Random Forest* dengan seleksi fitur yang dapat dilihat pada tabel 4.3.

Tabel 4 3 Waktu Pengerjaan

Nama Algoritme	Waktu Menyesuaikan	Waktu Menghasilkan Nilai
C45	0,003 detik	0,002 detik
C45-FS	0,002 detik	0,002 detik
RF	0,089 detik	0,006 detik
RF-FS	0,086 detik	0,005 detik

Nilai rata-rata waktu mengerjakan hasil pengujian dapat ditampilkan dalam bentuk grafik yang dapat dilihat pada gambar 4.16.



Gambar 4. 16 Grafik Perbandingan Rata-rata Waktu Mengerjakan Hasil Pengujian

Dari table dan grafik waktu pengerjaan terlihat adanya pengurangan waktu sesudah dilakukan seleksi fitur, karena data yang diolah hanya 400 record, sehingga selisih waktu pengerjaan cukup kecil, ini memungkinkan perbedaan waktu pengerjaan cukup besar setelah seleksi fitur jika data yang diolah menggunakan data yang besar.