DATA MINING

ANALISIS DAN PENGEMBANGAN APLIKASI DETEKSI PENYAKIT GINJAL KRONIS MENGGUNAKAN DATASET DARI KAGGLE

Disusun Oleh:

Hairul Anwar	105841109121
Aulya Agustin	105841108921
Muh. Reyhan	105841106921

Pendahuluan

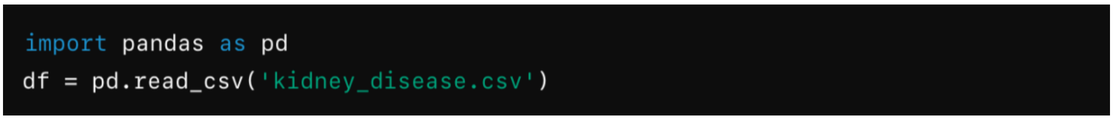
Penyakit Ginjal Kronis (Chronic Kidney Disease atau CKD) adalah masalah kesehatan global yang memerlukan perhatian khusus. Dalam mata kuliah Data Mining, kami menggunakan dataset CKD dari Kaggle untuk membangun aplikasi deteksi penyakit ini menggunakan Python.

Deskripsi Dataset

Dataset CKD dari Kaggle berisi 400 data pasien dengan 25 fitur medis, termasuk tekanan darah, kadar glukosa, hemoglobin, dan hasil tes lainnya. Kolom 'class' menunjukkan status CKD pasien (positif atau negatif).

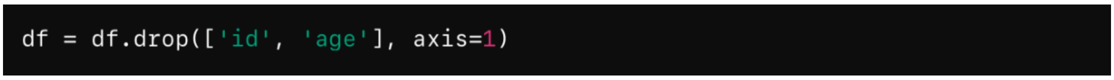
Tahap Pembersihan Data

1. Memuat Dataset: Dataset dimuat menggunakan pandas.



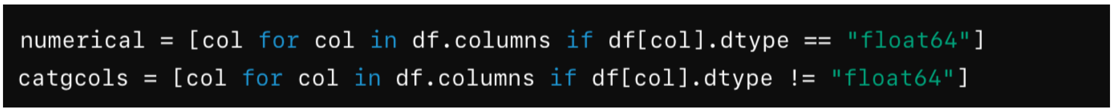
```python
import pandas as pd
df = pd.read_csv('kidney_disease.csv')
```

2. Menghapus Kolom Tidak Relevan: Kolom 'id' dan 'age' dihapus.



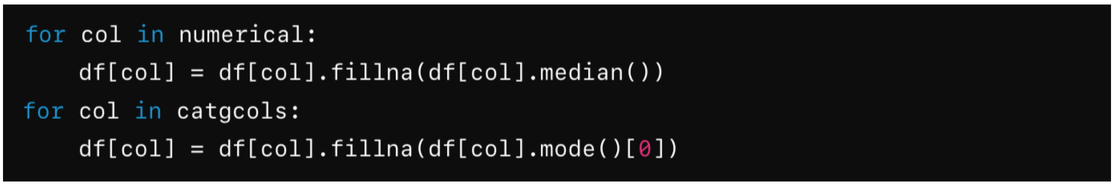
```python
df = df.drop(['id', 'age'], axis=1)
```

3. Mengidentifikasi Kolom Numerik dan Kategorikal: Kolom-kolom diidentifikasi berdasarkan tipe datanya.



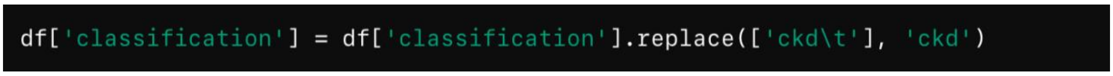
```python
numerical = [col for col in df.columns if df[col].dtype == "float64"]
catgcols = [col for col in df.columns if df[col].dtype != "float64"]
```

4. Mengisi Nilai Hilang: Nilai hilang diisi dengan median untuk kolom numerik dan modus untuk kolom kategorikal.



```python
for col in numerical:
    df[col] = df[col].fillna(df[col].median())
for col in catgcols:
    df[col] = df[col].fillna(df[col].mode()[0])
```

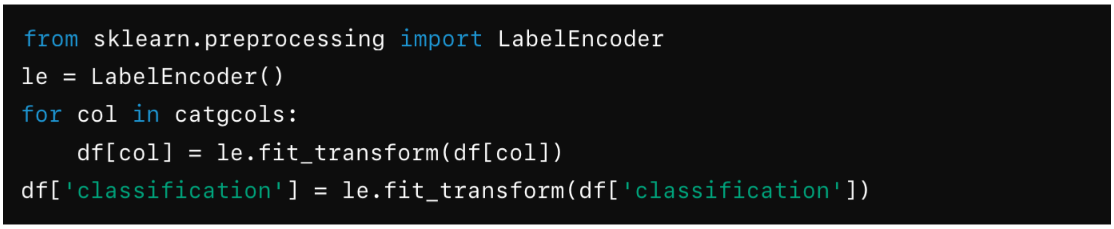
5. Membersihkan Label Klasifikasi: Label 'ckd\t' diganti menjadi 'ckd'.



```
df['classification'] = df['classification'].replace(['ckd\t'], 'ckd')
```

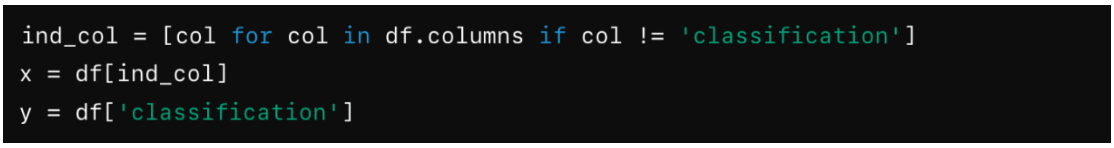
Transformasi Data

1. Encoding Data Kategorikal: Mengubah data kategorikal menjadi numerik menggunakan LabelEncoder.



```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for col in catgcols:
    df[col] = le.fit_transform(df[col])
df['classification'] = le.fit_transform(df['classification'])
```

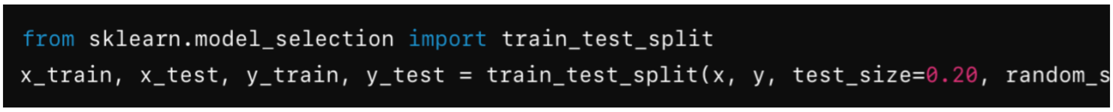
2. Memisahkan Fitur dan Label: Memisahkan kolom fitur dan label.



```
ind_col = [col for col in df.columns if col != 'classification']
x = df[ind_col]
y = df['classification']
```

Pembagian Dataset

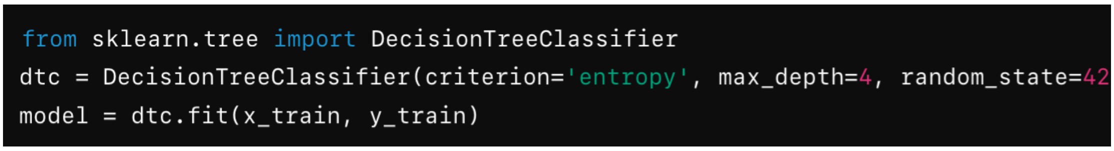
Dataset dibagi menjadi data pelatihan dan data pengujian



```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_s
```

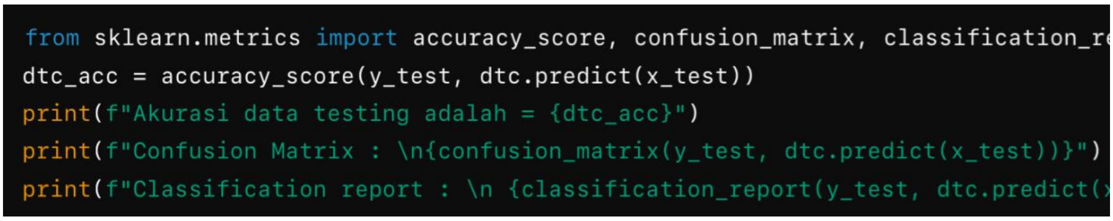
Pembangunan Model

1. Menggunakan Decision Tree Classifier: Model dibuat dan dilatih menggunakan Decision Tree.



```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier(criterion='entropy', max_depth=4, random_state=42
model = dtc.fit(x_train, y_train)
```

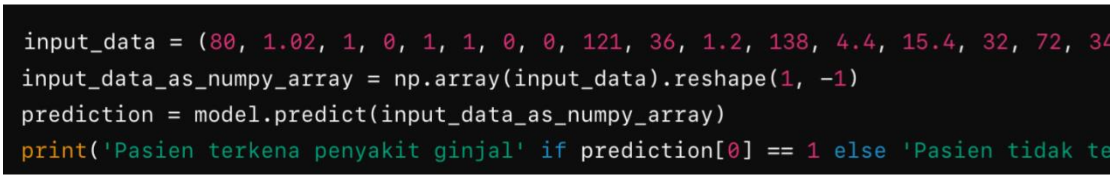
2. Evaluasi Model: Model dievaluasi menggunakan akurasi, confusion matrix, dan classification report.



```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_r
dtc_acc = accuracy_score(y_test, dtc.predict(x_test))
print(f"Akurasi data testing adalah = {dtc_acc}")
print(f"Confusion Matrix : \n{confusion_matrix(y_test, dtc.predict(x_test))}")
print(f"Classification report : \n {classification_report(y_test, dtc.predict(
```

Simulasi Model

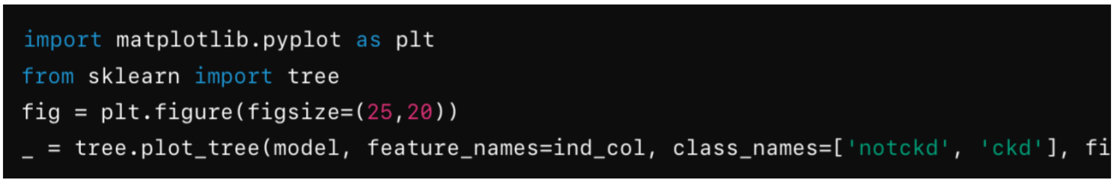
Simulasi prediksi dengan data input



```
input_data = (80, 1.02, 1, 0, 1, 1, 0, 0, 121, 36, 1.2, 138, 4.4, 15.4, 32, 72, 34
input_data_as_numpy_array = np.array(input_data).reshape(1, -1)
prediction = model.predict(input_data_as_numpy_array)
print('Pasien terkena penyakit ginjal' if prediction[0] == 1 else 'Pasien tidak te
```

Visualisasi Pohon Keputusan

Menampilkan pohon keputusan menggunakan Matplotib

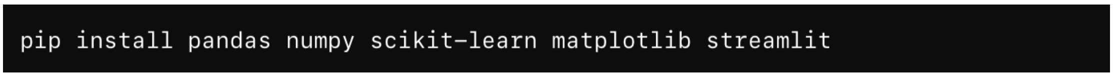


```
import matplotlib.pyplot as plt
from sklearn import tree
fig = plt.figure(figsize=(25,20))
_ = tree.plot_tree(model, feature_names=ind_col, class_names=['notckd', 'ckd'], fi
```

Pengembangan Aplikasi

Aplikasi ini dikembangkan menggunakan Python dengan antarmuka berbasis Streamlit. Berikut adalah langkah-langkah pengembangannya:

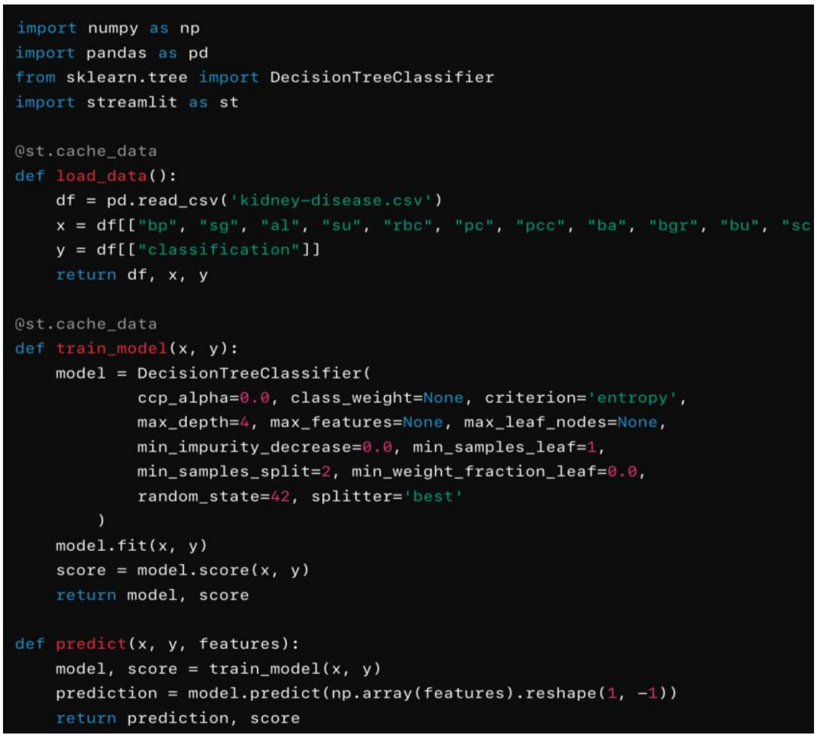
1. Menginstal Library yang digunakan



```
pip install pandas numpy scikit-learn matplotlib streamlit
```

2. Membangun Fungsi Web untuk Streamlit

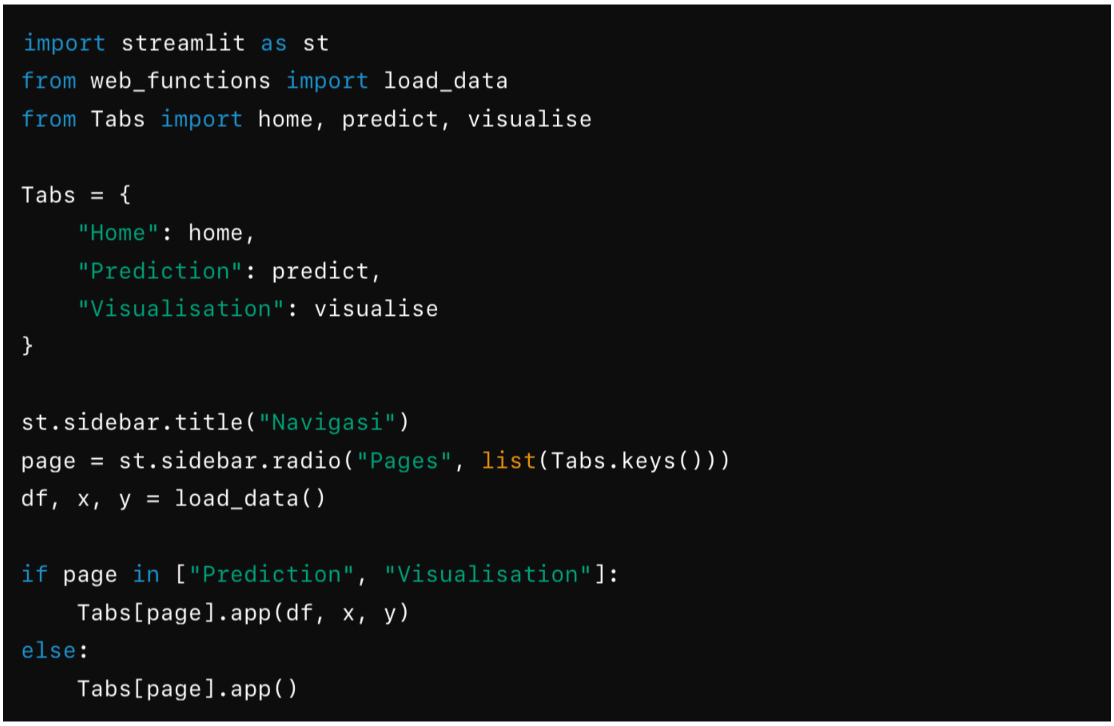
   File: ‘web_function.py’



```
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
import streamlit as st

@st.cache_data
def load_data():
    df = pd.read_csv('kidney-disease.csv')
    x = df[["bp", "sg", "al", "su", "rbc", "pc", "pcc", "ba", "bgr", "bu", "sc
    y = df[["classification"]]
    return df, x, y

@st.cache_data
def train_model(x, y):
    model = DecisionTreeClassifier(
            ccp_alpha=0.0, class_weight=None, criterion='entropy',
            max_depth=4, max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            random_state=42, splitter='best'
        )
    model.fit(x, y)
    score = model.score(x, y)
    return model, score

def predict(x, y, features):
    model, score = train_model(x, y)
    prediction = model.predict(np.array(features).reshape(1, -1))
    return prediction, score
```

3. Membangun File Utama Aplikasi

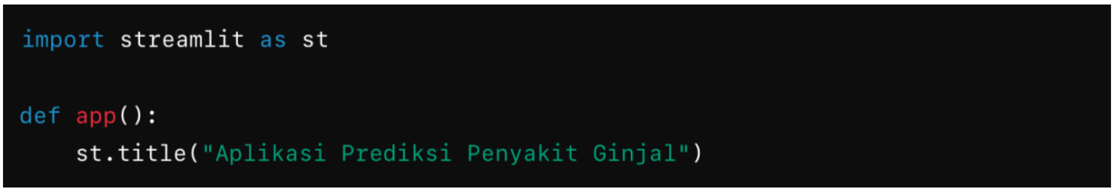
   File: ‘main.py’



```
import streamlit as st
from web_functions import load_data
from Tabs import home, predict, visualise

Tabs = {
    "Home": home,
    "Prediction": predict,
    "Visualisation": visualise
}

st.sidebar.title("Navigasi")
page = st.sidebar.radio("Pages", list(Tabs.keys()))
df, x, y = load_data()

if page in ["Prediction", "Visualisation"]:
    Tabs[page].app(df, x, y)
else:
    Tabs[page].app()
```

4. Membuat Halaman Beranda

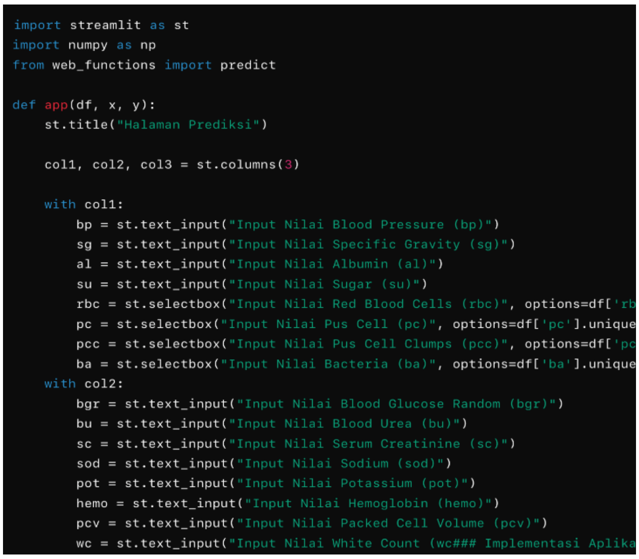
   File: ‘home.py



```
import streamlit as st

def app():
    st.title("Aplikasi Prediksi Penyakit Ginjal")
```

5. Membangun Halaman Prediksi

   File: ‘predict.py’



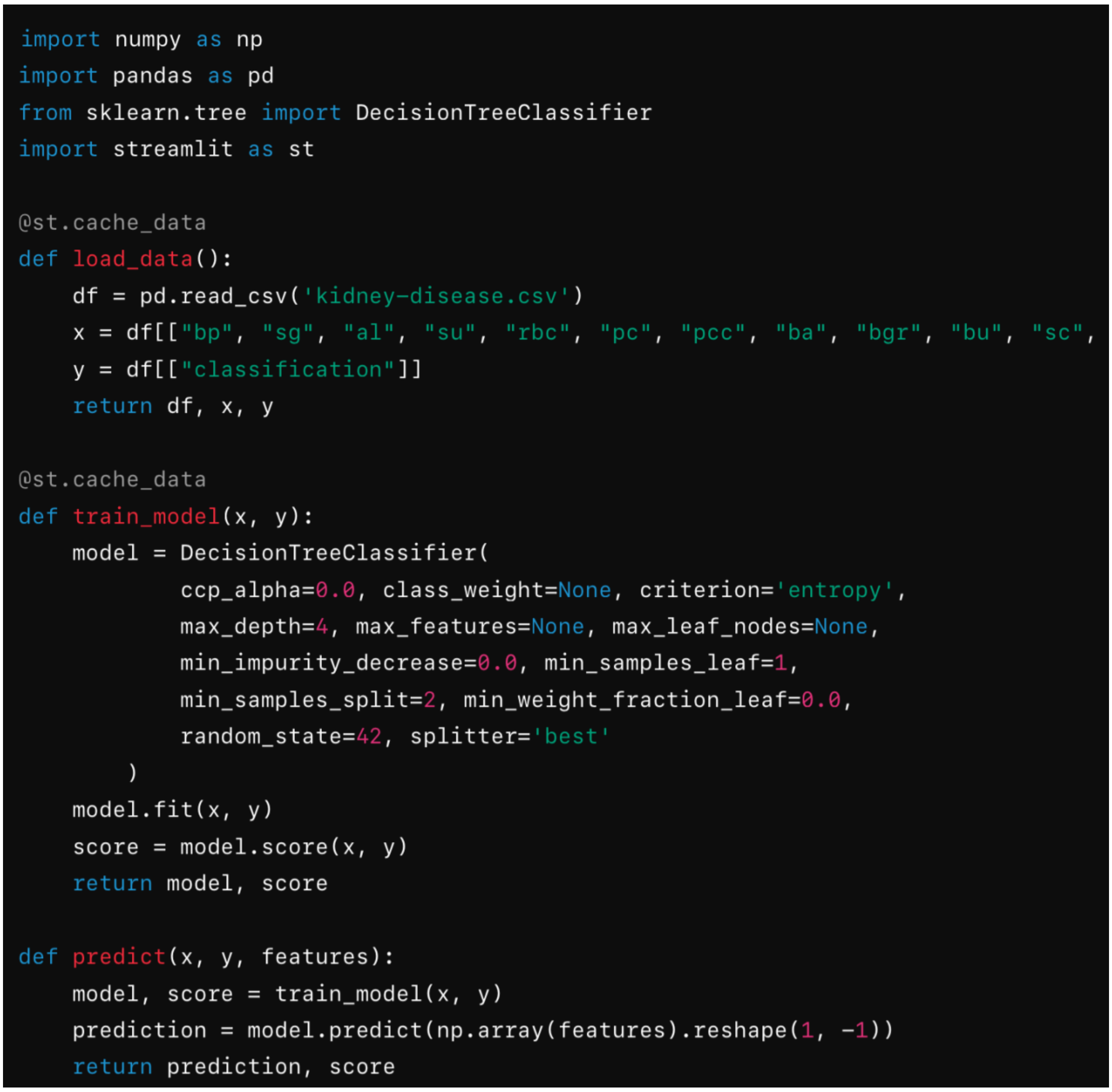
```
import streamlit as st
import numpy as np
from web_functions import predict

def app(df, x, y):
    st.title("Halaman Prediksi")

    col1, col2, col3 = st.columns(3)

    with col1:
        bp = st.text_input("Input Nilai Blood Pressure (bp)")
        sg = st.text_input("Input Nilai Specific Gravity (sg)")
        al = st.text_input("Input Nilai Albumin (al)")
        su = st.text_input("Input Nilai Sugar (su)")
        rbc = st.selectbox("Input Nilai Red Blood Cells (rbc)", options=df['rb
        pc = st.selectbox("Input Nilai Pus Cell (pc)", options=df['pc'].unique
        pcc = st.selectbox("Input Nilai Pus Cell Clumps (pcc)", options=df['pc
        ba = st.selectbox("Input Nilai Bacteria (ba)", options=df['ba'].unique
    with col2:
        bgr = st.text_input("Input Nilai Blood Glucose Random (bgr)")
        bu = st.text_input("Input Nilai Blood Urea (bu)")
        sc = st.text_input("Input Nilai Serum Creatinine (sc)")
        sod = st.text_input("Input Nilai Sodium (sod)")
        pot = st.text_input("Input Nilai Potassium (pot)")
        hemo = st.text_input("Input Nilai Hemoglobin (hemo)")
        pcv = st.text_input("Input Nilai Packed Cell Volume (pcv)")
        wc = st.text_input("Input Nilai White Count (wc### Implementasi Aplika
```

Deskripsi

Aplikasi ini dikembangkan menggunakan Python dan Streamlit untuk memprediksi penyakit ginjal kronis berdasarkan data medis. Berikut adalah langkah-langkah pengembangannya.

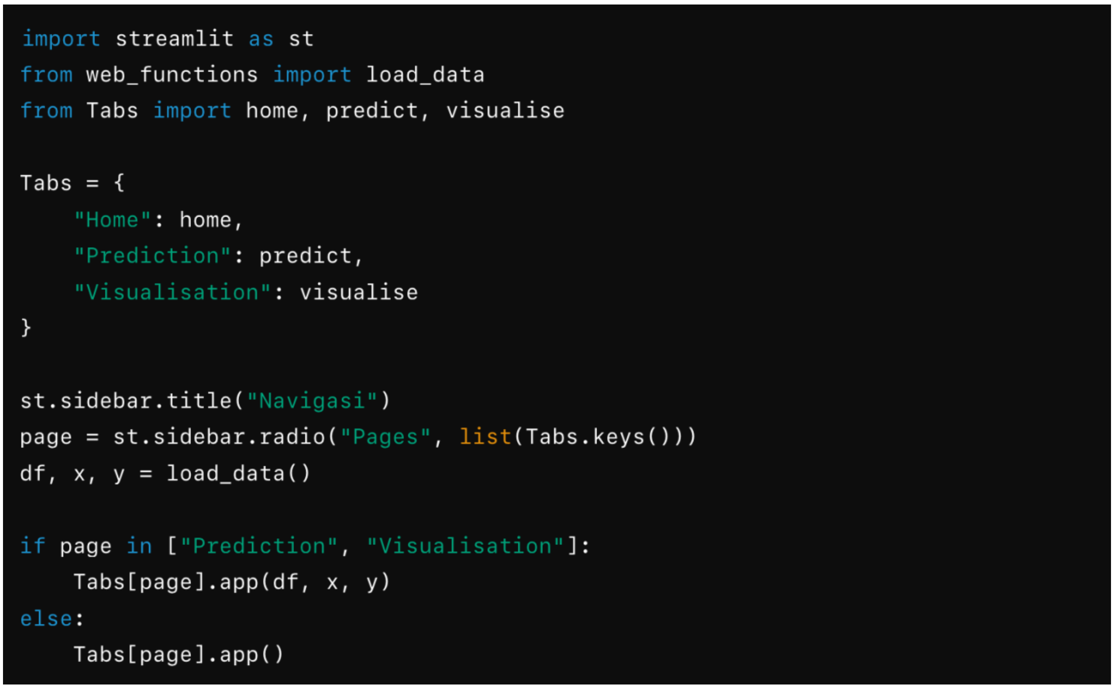
Struktur Data

1. File web_function.py: Berisi fungsi utama untuk memuat data, melatih model, dan memprediksi hasil.



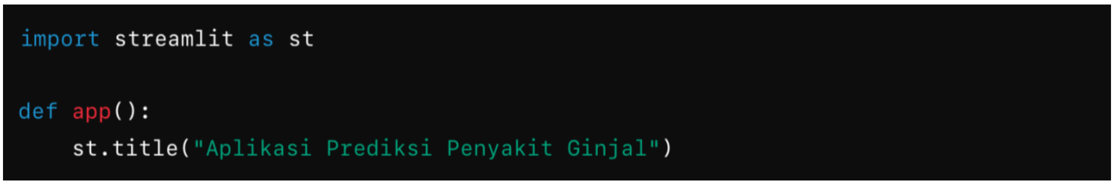
```
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
import streamlit as st

@st.cache_data
def load_data():
    df = pd.read_csv('kidney-disease.csv')
    x = df[["bp", "sg", "al", "su", "rbc", "pc", "pcc", "ba", "bgr", "bu", "sc",
    y = df[["classification"]]
    return df, x, y

@st.cache_data
def train_model(x, y):
    model = DecisionTreeClassifier(
            ccp_alpha=0.0, class_weight=None, criterion='entropy',
            max_depth=4, max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            random_state=42, splitter='best'
        )
    model.fit(x, y)
    score = model.score(x, y)
    return model, score

def predict(x, y, features):
    model, score = train_model(x, y)
    prediction = model.predict(np.array(features).reshape(1, -1))
    return prediction, score
```

2. File main.py: Berisi logika utama aplikasi dan navigasi antar halaman.



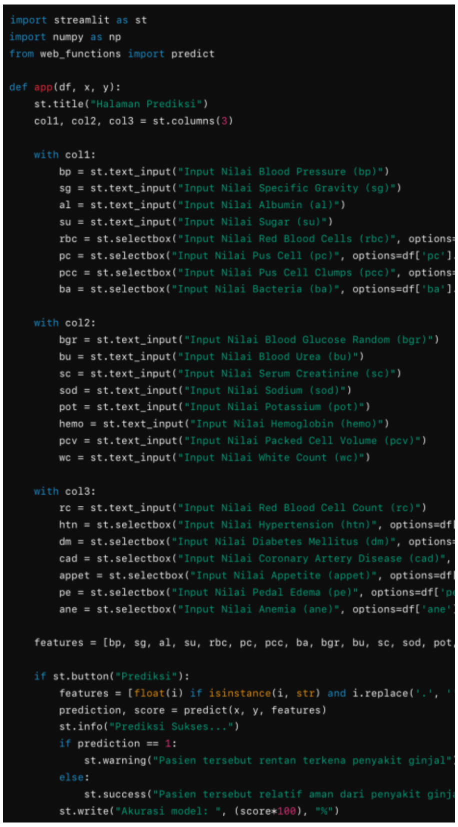
```python
import streamlit as st
from web_functions import load_data
from Tabs import home, predict, visualise

Tabs = {
    "Home": home,
    "Prediction": predict,
    "Visualisation": visualise
}

st.sidebar.title("Navigasi")
page = st.sidebar.radio("Pages", list(Tabs.keys()))
df, x, y = load_data()

if page in ["Prediction", "Visualisation"]:
    Tabs[page].app(df, x, y)
else:
    Tabs[page].app()
```

3. File home.py: Berisi halaman utama aplikasi.



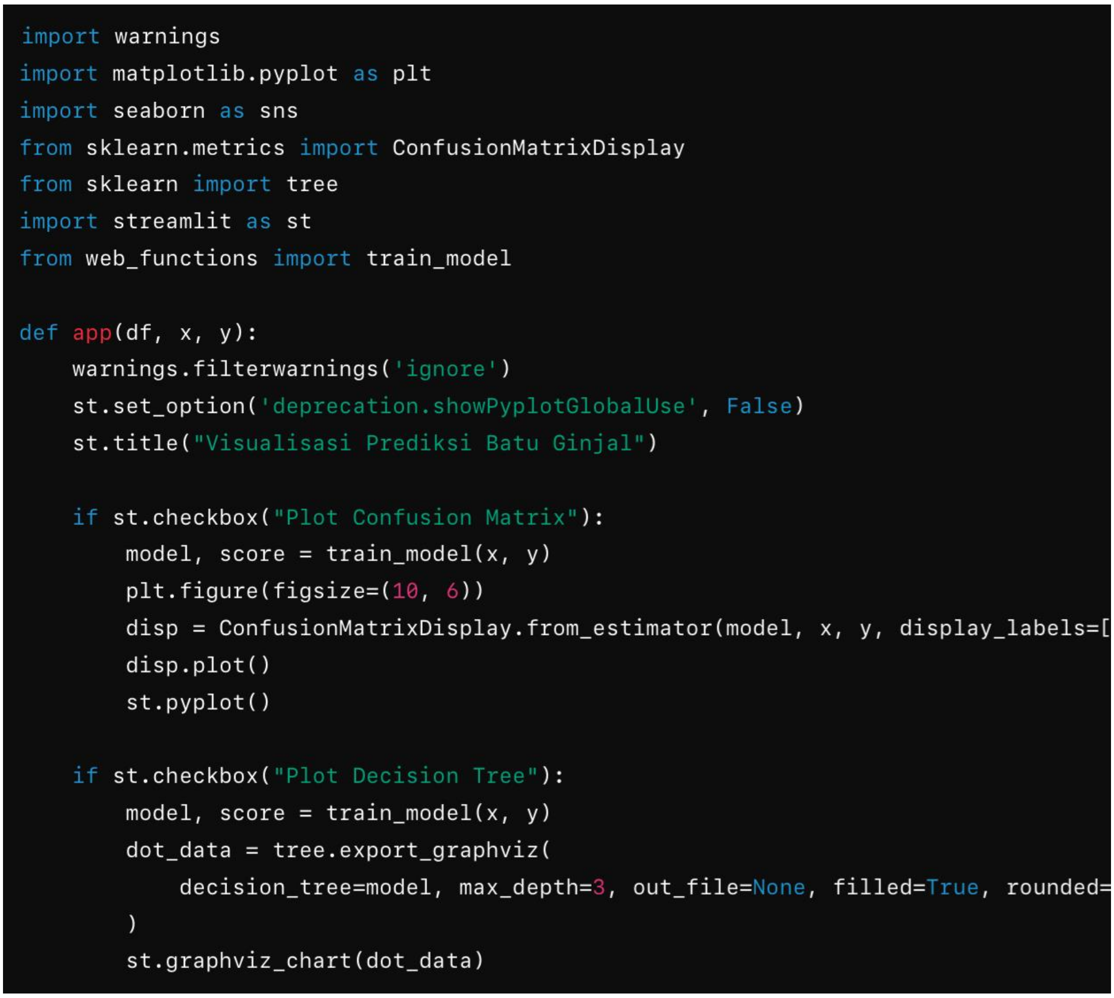
```python
import streamlit as st

def app():
    st.title("Aplikasi Prediksi Penyakit Ginjal")
```

4. File predict.py: Berisi logika prediksi dan tampilan input fitur.



```python
import streamlit as st
import numpy as np
from web_functions import predict

def app(df, x, y):
    st.title("Halaman Prediksi")
    col1, col2, col3 = st.columns(3)

    with col1:
        bp = st.text_input("Input Nilai Blood Pressure (bp)")
        sg = st.text_input("Input Nilai Specific Gravity (sg)")
        al = st.text_input("Input Nilai Albumin (al)")
        su = st.text_input("Input Nilai Sugar (su)")
        rbc = st.selectbox("Input Nilai Red Blood Cells (rbc)", options=
        pc = st.selectbox("Input Nilai Pus Cell (pc)", options=df['pc'].
        pcc = st.selectbox("Input Nilai Pus Cell Clumps (pcc)", options=
        ba = st.selectbox("Input Nilai Bacteria (ba)", options=df['ba'].

    with col2:
        bgr = st.text_input("Input Nilai Blood Glucose Random (bgr)")
        bu = st.text_input("Input Nilai Blood Urea (bu)")
        sc = st.text_input("Input Nilai Serum Creatinine (sc)")
        sod = st.text_input("Input Nilai Sodium (sod)")
        pot = st.text_input("Input Nilai Potassium (pot)")
        hemo = st.text_input("Input Nilai Hemoglobin (hemo)")
        pcv = st.text_input("Input Nilai Packed Cell Volume (pcv)")
        wc = st.text_input("Input Nilai White Count (wc)")

    with col3:
        rc = st.text_input("Input Nilai Red Blood Cell Count (rc)")
        htn = st.selectbox("Input Nilai Hypertension (htn)", options=df[
        dm = st.selectbox("Input Nilai Diabetes Mellitus (dm)", options=
        cad = st.selectbox("Input Nilai Coronary Artery Disease (cad)",
        appet = st.selectbox("Input Nilai Appetite (appet)", options=df[
        pe = st.selectbox("Input Nilai Pedal Edema (pe)", options=df['pe
        ane = st.selectbox("Input Nilai Anemia (ane)", options=df['ane']

    features = [bp, sg, al, su, rbc, pc, pcc, ba, bgr, bu, sc, sod, pot,

    if st.button("Prediksi"):
        features = [float(i) if isinstance(i, str) and i.replace('.', '
        prediction, score = predict(x, y, features)
        st.info("Prediksi Sukses...")
        if prediction == 1:
            st.warning("Pasien tersebut rentan terkena penyakit ginjal")
        else:
            st.success("Pasien tersebut relatif aman dari penyakit ginj
        st.write("Akurasi model: ", (score*100), "%")
```

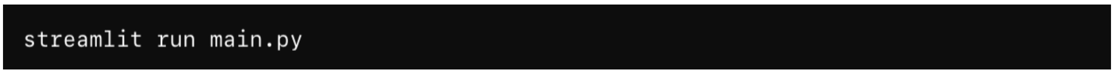
5. File visualise.py: Berisi visualisasi hasil prediksi dan pohon keputusan.



```python
import warnings
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn import tree
import streamlit as st
from web_functions import train_model

def app(df, x, y):
    warnings.filterwarnings('ignore')
    st.set_option('deprecation.showPyplotGlobalUse', False)
    st.title("Visualisasi Prediksi Batu Ginjal")

    if st.checkbox("Plot Confusion Matrix"):
        model, score = train_model(x, y)
        plt.figure(figsize=(10, 6))
        disp = ConfusionMatrixDisplay.from_estimator(model, x, y, display_labels=[
        disp.plot()
        st.pyplot()

    if st.checkbox("Plot Decision Tree"):
        model, score = train_model(x, y)
        dot_data = tree.export_graphviz(
            decision_tree=model, max_depth=3, out_file=None, filled=True, rounded=
        )
        st.graphviz_chart(dot_data)
```

Menjalankan Aplikasi

Jalankan aplikasi menggunakan Streamlit diterminal atau Comand Promt



```
streamlit run main.py
```

Aplikasi ini akan menampilkan antarmuka web untuk prediksi penyakit ginjal berdasarkan input fitur medis, menampilkan akurasi model, dan visualisasi pohon keputusan serta confusion matrix.