Nama Dosen : Teguh Iman Hermanto, M.Kom
Mata Kuliah : Machine Learning 1
Pembahasan : Naïve Bayes
Pokok Pemb : - Konsep Algoritma Naïve Bayes
- Membangun Model Naïve Bayes
- Simulasi Algoritma Naïve Bayes
- Evaluasi Algoritma Naïve Bayes
- Aplikasi menggunakan algoritma Naïve Bayes

1. Konsep Algoritma Naïve Bayes

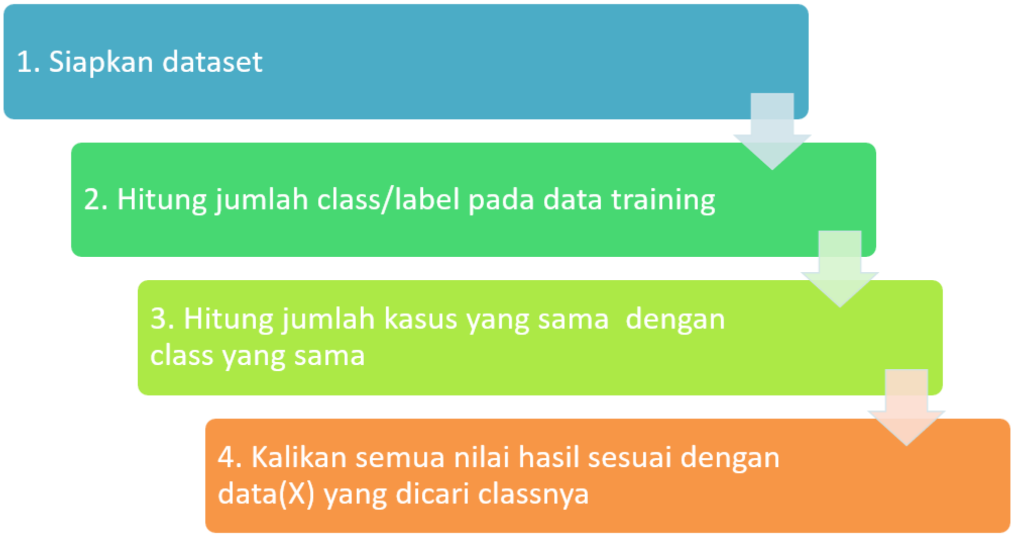
Naïve Bayes Classifier merupakan sebuah metoda klasifikasi yang berakar pada teorema Bayes . Metode pengklasifikasian dg menggunakan metode probabilitas dan statistik yg dikemukakan oleh ilmuwan Inggris Thomas Bayes , yaitu memprediksi peluang di masa depan berdasarkan pengalaman di masa sebelumnya sehingga dikenal sebagai Teorema Bayes . Ciri utama dr Naïve Bayes Classifier ini adalah asumsi yg sangat kuat (naïf) akan independensi dari masing-masing kondisi / kejadian.

Model Algoritma Naïve Bayes

$$P(H|X)=\frac{P(X|H)P(H)}{P(X)}=P(X|H)\times P(H)/P(X)$$

x = Data dari class scoring yang belum diketahui
H = Hipotesis data X yang merupakan suatu class yang lebih spesifik
P(H|X) = Probabilitas hipotesis H berdasarkan kondisi X
P(H) = Probabilitas hipotesis H
P(X|H) = Probabilitas hipotesis X berdasarkan kondisi H
P(X) = Probabilitas X

Tahapan Algoritma Naïve Bayes



1. Siapkan dataset

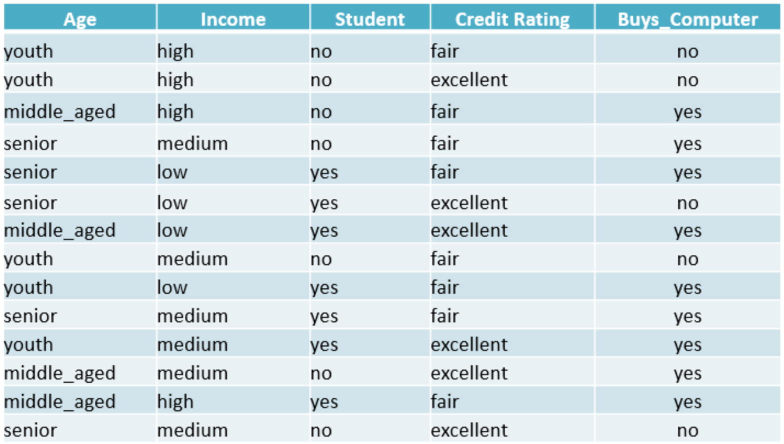
2. Hitung jumlah class/label pada data training

3. Hitung jumlah kasus yang sama dengan class yang sama

4. Kalikan semua nilai hasil sesuai dengan data(X) yang dicari classnya

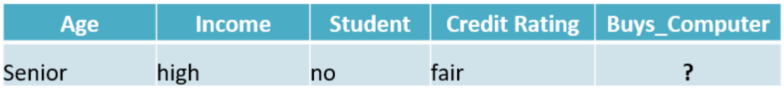
a. Siapkan Dataset

- Data training buys_computer dataset



Age	Income	Student	Credit Rating	Buys_Computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

- Data testing



Age	Income	Student	Credit Rating	Buys_Computer
Senior	high	no	fair	?

Apakah calon pembeli akan membeli komputer?

b. Hitung jumlah class/label pada data training

Class 1 (C1)→ Buys_Computer yes → 9

Class 2 (C2) → Buys_Computer no → 5

Total Record: 14

Maka:

P(C1) = 9/14 = 0.642857143

P(C2) = 5/14 = 0.357142857

c. Hitung jumlah kasus yang sama dengan class yang sama

Contoh penghitungan P(X|Ci) pada kasus Income dimana

i=1 dan i=2

Maka :

P(Income="high"|Buys_Computer="yes") = 2/9 = 0.22222222

P(Income="high"|Buys_Computer="no") = 2/5 = 0.4

d. Kalikan semua nilai hasil sesuai dengan data(X) yang dicari classnya

Jika semua data aribut pada data training dihitung, maka:

Atribut	Parameter	yes	no
Age	youth	0.222222222	0.6
Age	middle_aged	0.444444444	0
Age	**senior**	**0.333333333**	**0.4**
Income	low	0.333333333	0.2
Income	medium	0.444444444	0.4
Income	**high**	**0.222222222**	**0.4**
Student	yes	0.666666667	0.2
Student	**no**	**0.333333333**	**0.8**
Credit Rating	**fair**	**0.666666667**	**0.4**
Credit Rating	excellent	0.333333333	0.6

P(X|Buys_Computer="yes") = 0.016460905

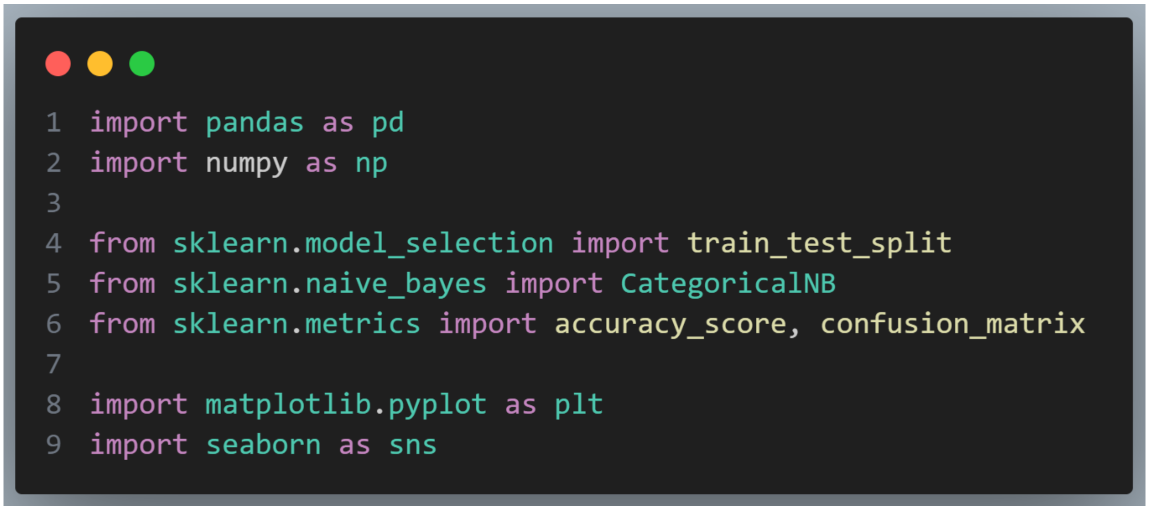
P(X|Buys_Computer="no") = 0.0512

P(X|Buys_Computer="yes")*P(C1) = 0.010582011

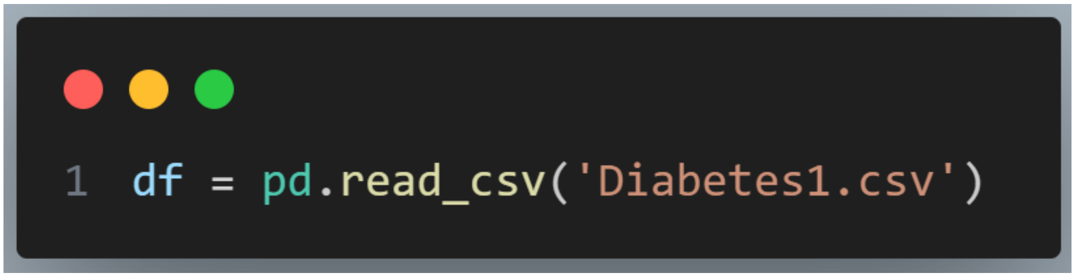
P(X|Buys_Computer="no")*P(C2) = 0.018285714

Nilai "no" lebih besar daripada nilai "yes". Sehingga pembeli dengan atribut X tidak membeli komputer

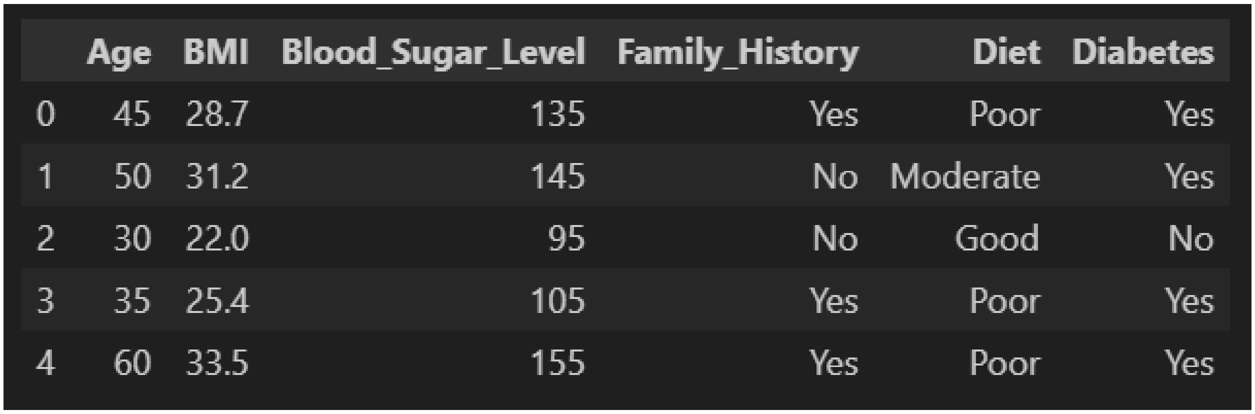
2. Membangun Model Naïve Bayes



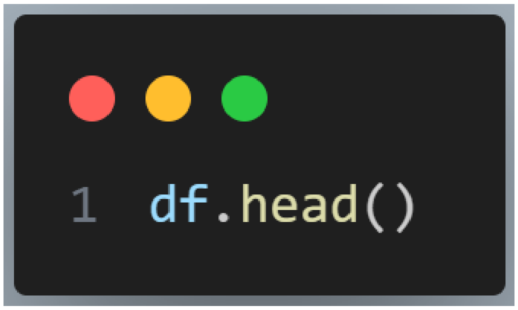
```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import CategoricalNB
from sklearn.metrics import accuracy_score, confusion_matrix

import matplotlib.pyplot as plt
import seaborn as sns
```



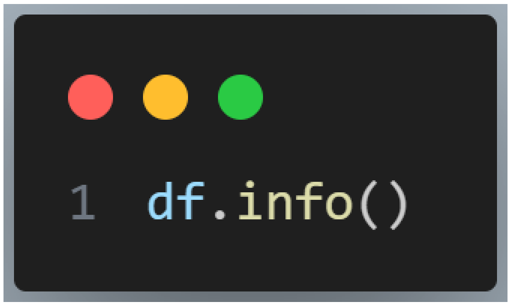
```python
df = pd.read_csv('Diabetes1.csv')
```



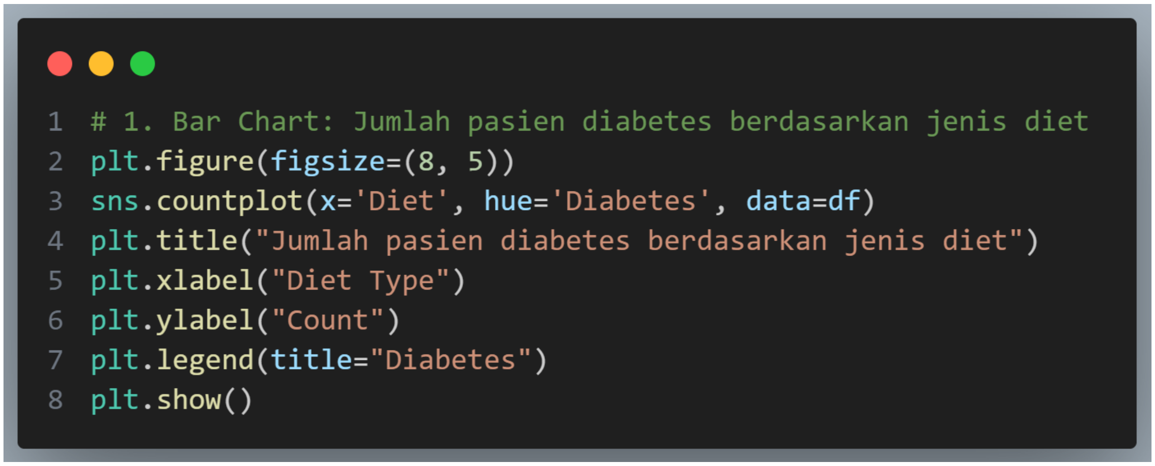
	Age	BMI	Blood_Sugar_Level	Family_History	Diet	Diabetes
0	45	28.7	135	Yes	Poor	Yes
1	50	31.2	145	No	Moderate	Yes
2	30	22.0	95	No	Good	No
3	35	25.4	105	Yes	Poor	Yes
4	60	33.5	155	Yes	Poor	Yes



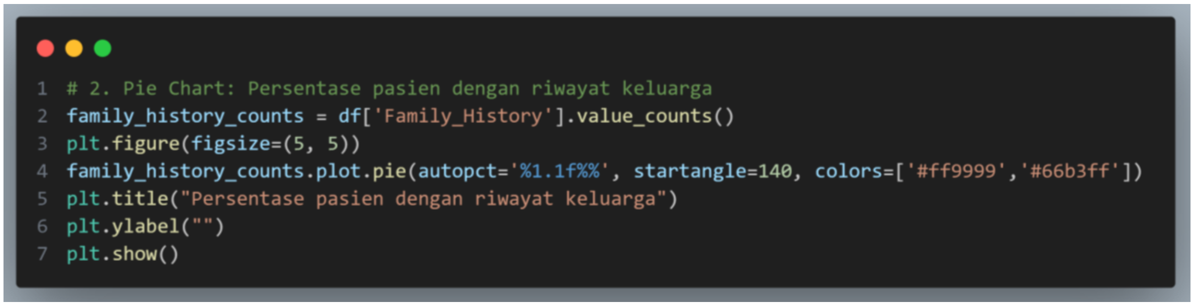
```python
df.head()
```



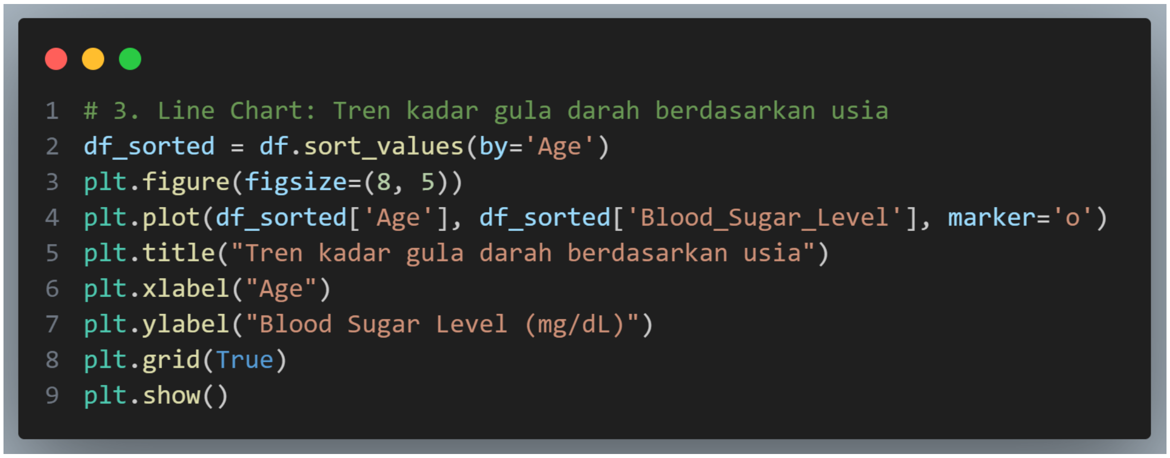
```python
df.info()
```



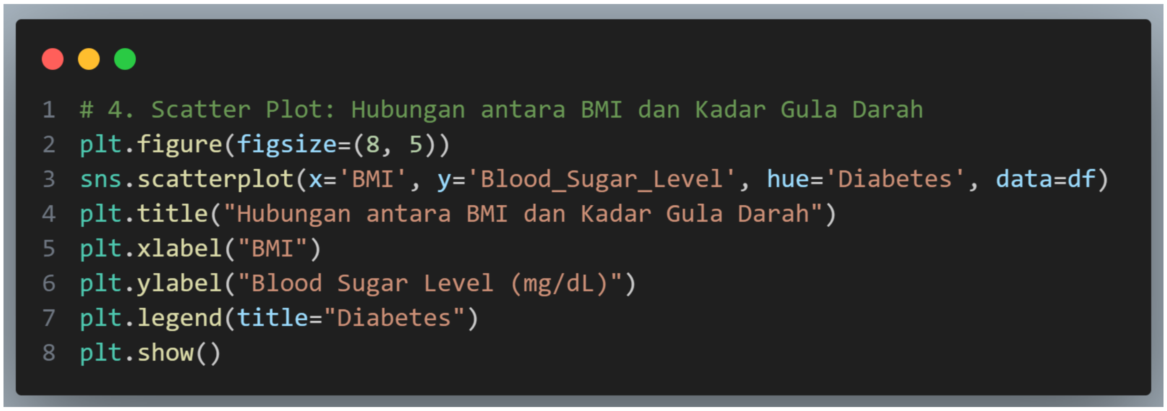
```python
# 1. Bar Chart: Jumlah pasien diabetes berdasarkan jenis diet
plt.figure(figsize=(8, 5))
sns.countplot(x='Diet', hue='Diabetes', data=df)
plt.title("Jumlah pasien diabetes berdasarkan jenis diet")
plt.xlabel("Diet Type")
plt.ylabel("Count")
plt.legend(title="Diabetes")
plt.show()
```



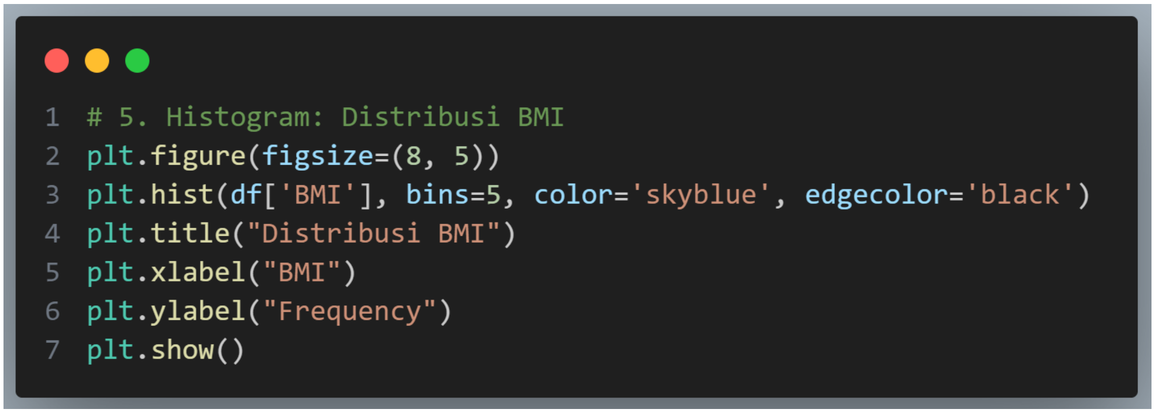
```python
# 2. Pie Chart: Persentase pasien dengan riwayat keluarga
family_history_counts = df['Family_History'].value_counts()
plt.figure(figsize=(5, 5))
family_history_counts.plot.pie(autopct='%1.1f%%', startangle=140, colors=['#ff9999','#66b3ff'])
plt.title("Persentase pasien dengan riwayat keluarga")
plt.ylabel("")
plt.show()
```



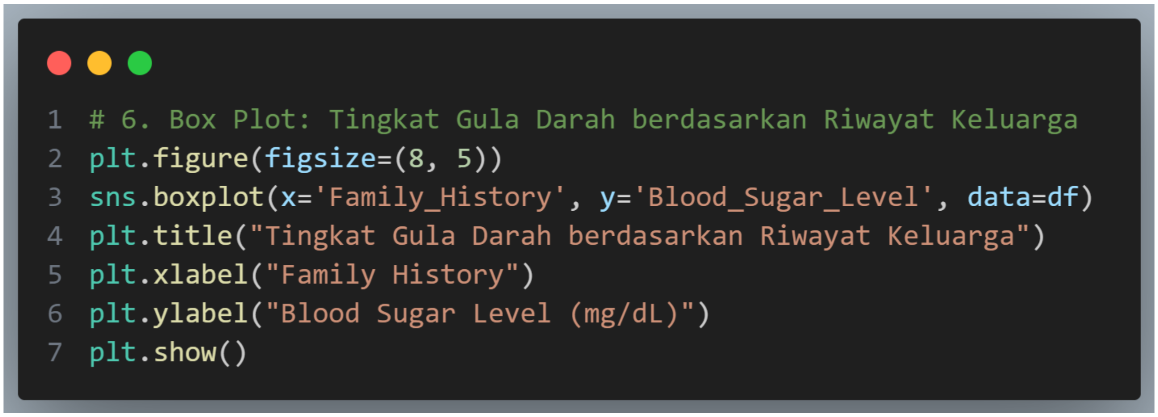
```python
# 3. Line Chart: Tren kadar gula darah berdasarkan usia
df_sorted = df.sort_values(by='Age')
plt.figure(figsize=(8, 5))
plt.plot(df_sorted['Age'], df_sorted['Blood_Sugar_Level'], marker='o')
plt.title("Tren kadar gula darah berdasarkan usia")
plt.xlabel("Age")
plt.ylabel("Blood Sugar Level (mg/dL)")
plt.grid(True)
plt.show()
```



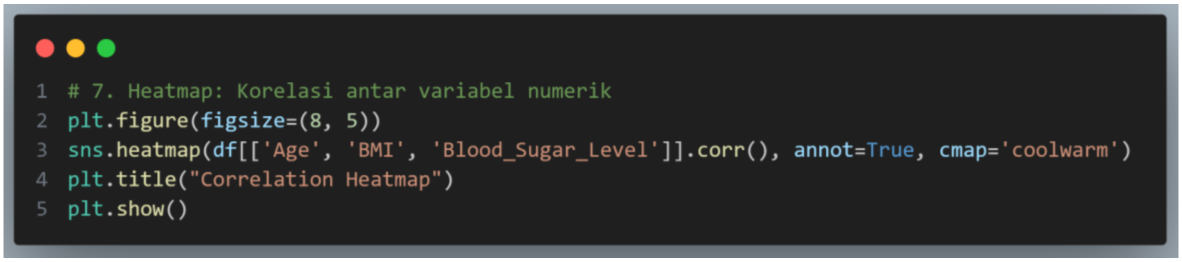
```python
# 4. Scatter Plot: Hubungan antara BMI dan Kadar Gula Darah
plt.figure(figsize=(8, 5))
sns.scatterplot(x='BMI', y='Blood_Sugar_Level', hue='Diabetes', data=df)
plt.title("Hubungan antara BMI dan Kadar Gula Darah")
plt.xlabel("BMI")
plt.ylabel("Blood Sugar Level (mg/dL)")
plt.legend(title="Diabetes")
plt.show()
```



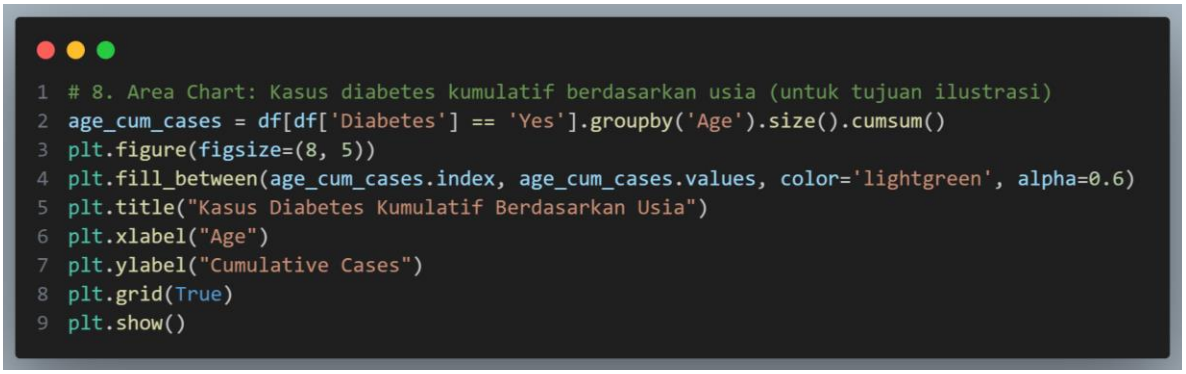
```python
# 5. Histogram: Distribusi BMI
plt.figure(figsize=(8, 5))
plt.hist(df['BMI'], bins=5, color='skyblue', edgecolor='black')
plt.title("Distribusi BMI")
plt.xlabel("BMI")
plt.ylabel("Frequency")
plt.show()
```



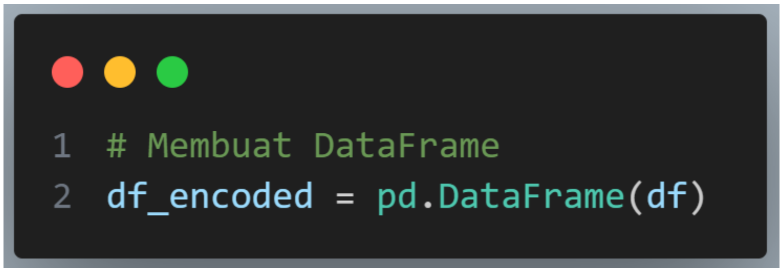
```python
# 6. Box Plot: Tingkat Gula Darah berdasarkan Riwayat Keluarga
plt.figure(figsize=(8, 5))
sns.boxplot(x='Family_History', y='Blood_Sugar_Level', data=df)
plt.title("Tingkat Gula Darah berdasarkan Riwayat Keluarga")
plt.xlabel("Family History")
plt.ylabel("Blood Sugar Level (mg/dL)")
plt.show()
```



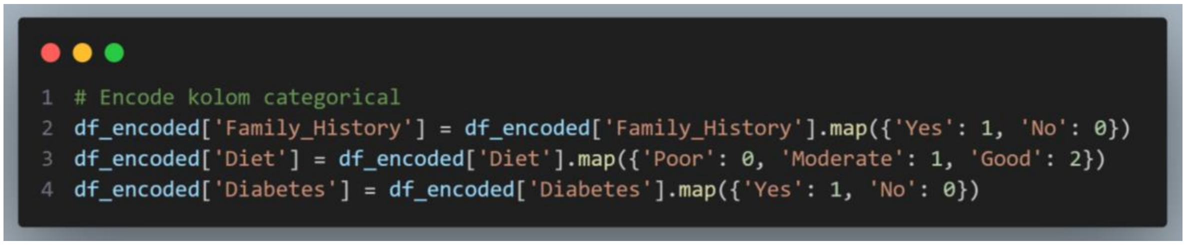
```python
# 7. Heatmap: Korelasi antar variabel numerik
plt.figure(figsize=(8, 5))
sns.heatmap(df[['Age', 'BMI', 'Blood_Sugar_Level']].corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()
```



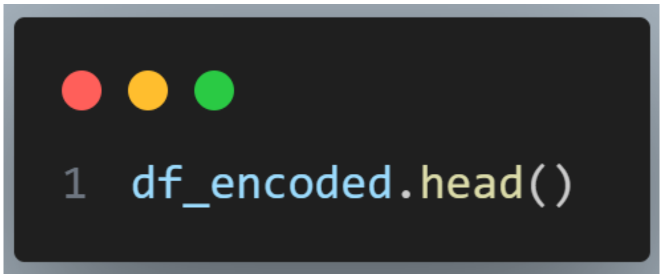
```python
# 8. Area Chart: Kasus diabetes kumulatif berdasarkan usia (untuk tujuan ilustrasi)
age_cum_cases = df[df['Diabetes'] == 'Yes'].groupby('Age').size().cumsum()
plt.figure(figsize=(8, 5))
plt.fill_between(age_cum_cases.index, age_cum_cases.values, color='lightgreen', alpha=0.6)
plt.title("Kasus Diabetes Kumulatif Berdasarkan Usia")
plt.xlabel("Age")
plt.ylabel("Cumulative Cases")
plt.grid(True)
plt.show()
```



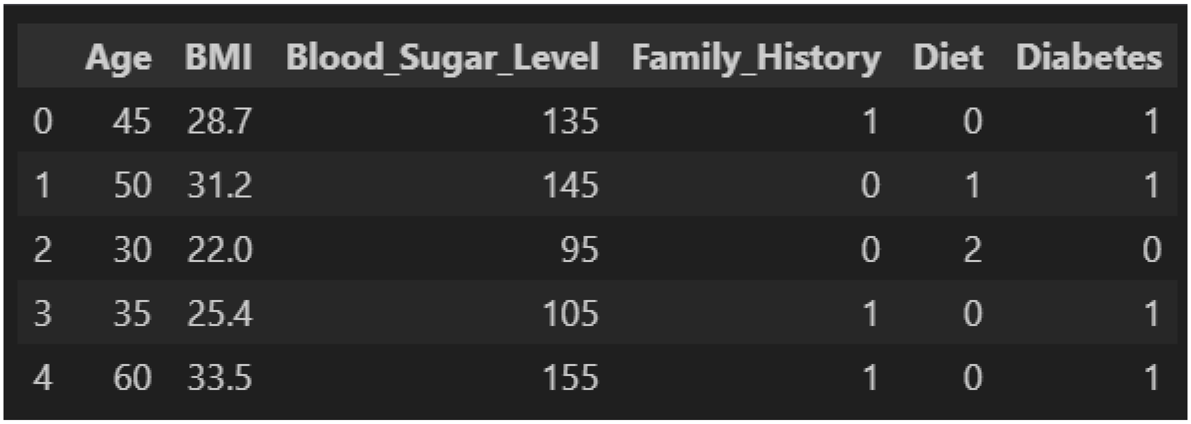
```python
# Membuat DataFrame
df_encoded = pd.DataFrame(df)
```



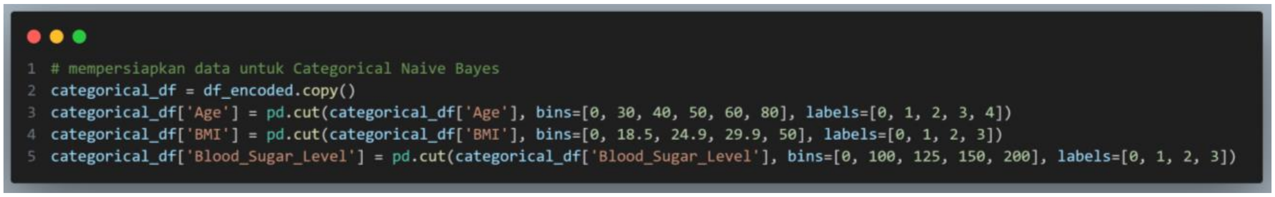
```python
# Encode kolom categorical
df_encoded['Family_History'] = df_encoded['Family_History'].map({'Yes': 1, 'No': 0})
df_encoded['Diet'] = df_encoded['Diet'].map({'Poor': 0, 'Moderate': 1, 'Good': 2})
df_encoded['Diabetes'] = df_encoded['Diabetes'].map({'Yes': 1, 'No': 0})
```



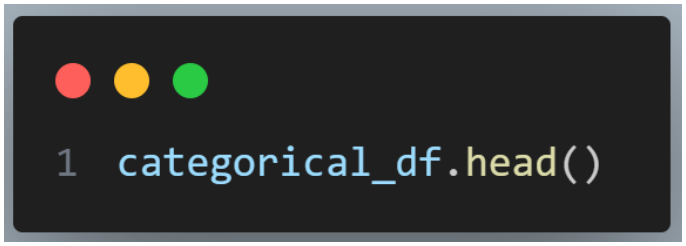
```python
df_encoded.head()
```



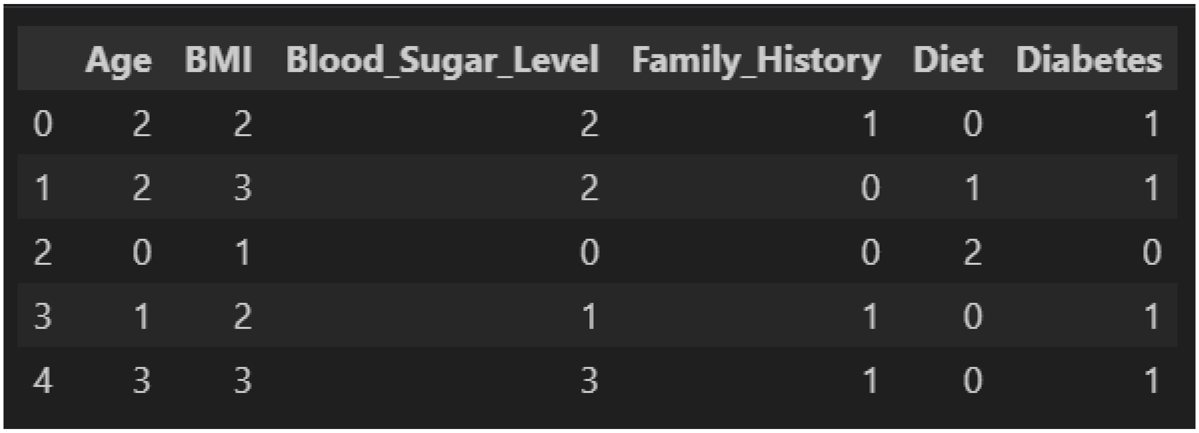
	Age	BMI	Blood_Sugar_Level	Family_History	Diet	Diabetes
0	45	28.7	135	1	0	1
1	50	31.2	145	0	1	1
2	30	22.0	95	0	2	0
3	35	25.4	105	1	0	1
4	60	33.5	155	1	0	1



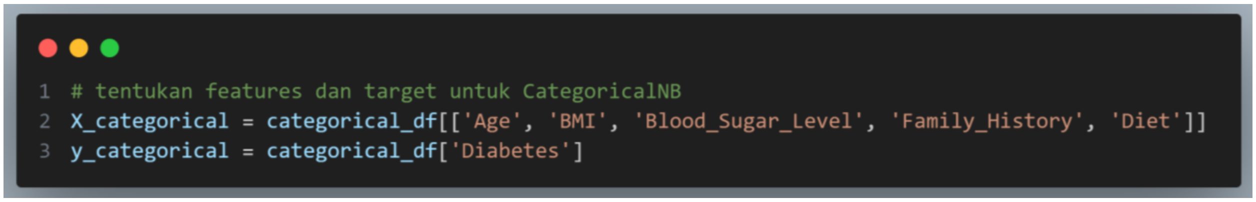
```python
# mempersiapkan data untuk Categorical Naive Bayes
categorical_df = df_encoded.copy()
categorical_df['Age'] = pd.cut(categorical_df['Age'], bins=[0, 30, 40, 50, 60, 80], labels=[0, 1, 2, 3, 4])
categorical_df['BMI'] = pd.cut(categorical_df['BMI'], bins=[0, 18.5, 24.9, 29.9, 50], labels=[0, 1, 2, 3])
categorical_df['Blood_Sugar_Level'] = pd.cut(categorical_df['Blood_Sugar_Level'], bins=[0, 100, 125, 150, 200], labels=[0, 1, 2, 3])
```



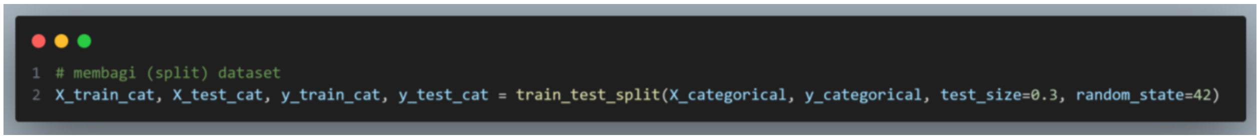
```python
categorical_df.head()
```



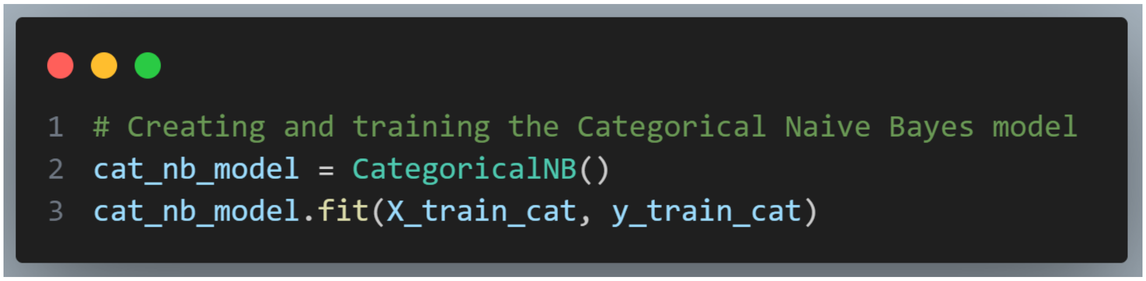
	Age	BMI	Blood_Sugar_Level	Family_History	Diet	Diabetes
0	2	2	2	1	0	1
1	2	3	2	0	1	1
2	0	1	0	0	2	0
3	1	2	1	1	0	1
4	3	3	3	1	0	1



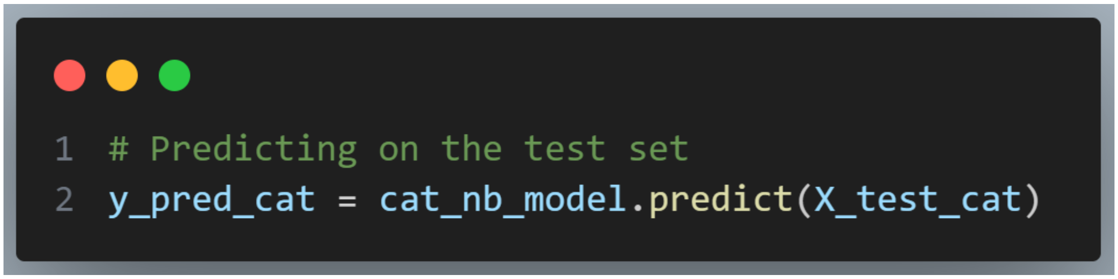
```
# tentukan features dan target untuk CategoricalNB
X_categorical = categorical_df[['Age', 'BMI', 'Blood_Sugar_Level', 'Family_History', 'Diet']]
y_categorical = categorical_df['Diabetes']
```



```
# membagi (split) dataset
X_train_cat, X_test_cat, y_train_cat, y_test_cat = train_test_split(X_categorical, y_categorical, test_size=0.3, random_state=42)
```

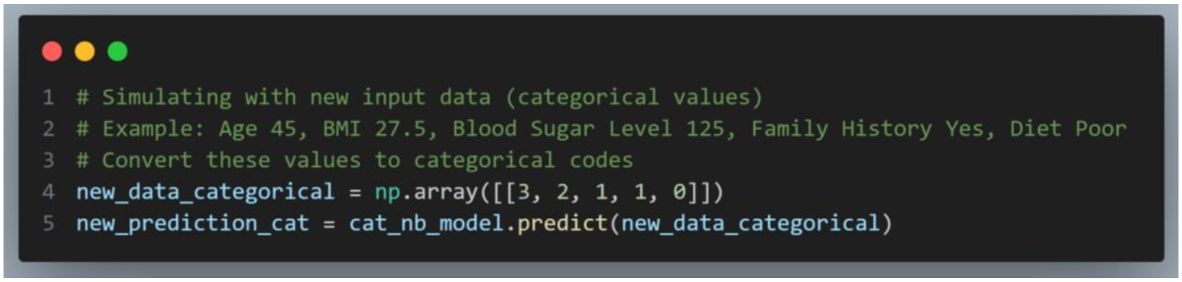


```
# Creating and training the Categorical Naive Bayes model
cat_nb_model = CategoricalNB()
cat_nb_model.fit(X_train_cat, y_train_cat)
```

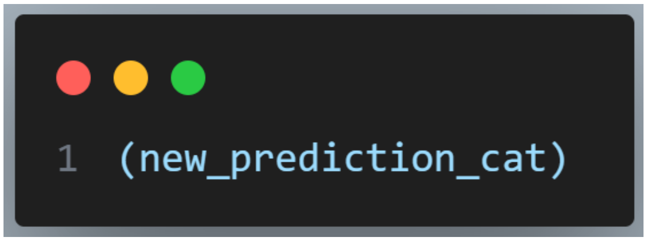


```
# Predicting on the test set
y_pred_cat = cat_nb_model.predict(X_test_cat)
```

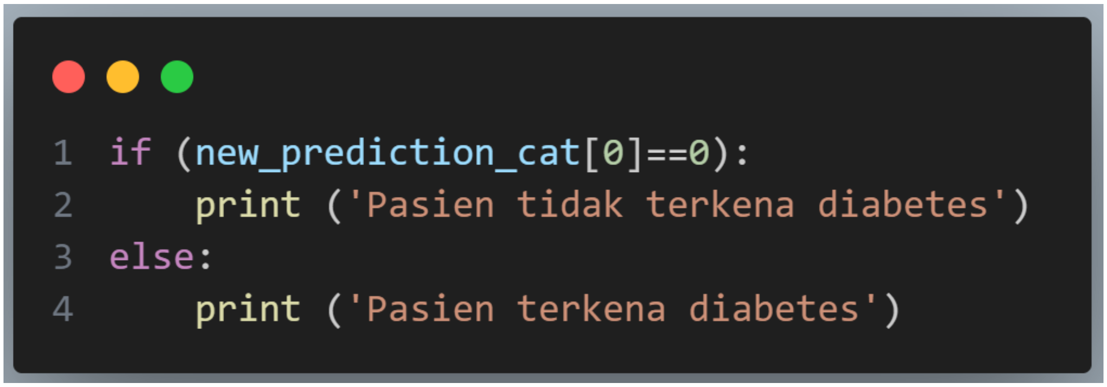
3. Simulasi Algoritma Naïve Bayes



```
# Simulating with new input data (categorical values)
# Example: Age 45, BMI 27.5, Blood Sugar Level 125, Family History Yes, Diet Poor
# Convert these values to categorical codes
new_data_categorical = np.array([[3, 2, 1, 1, 0]])
new_prediction_cat = cat_nb_model.predict(new_data_categorical)
```

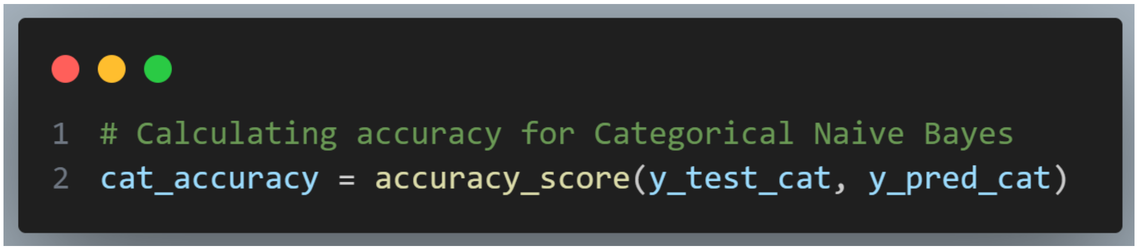


```
(new_prediction_cat)
```

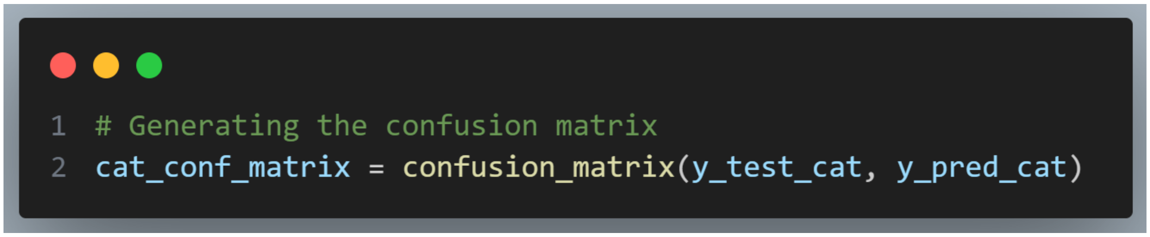


```
if (new_prediction_cat[0]==0):
      print ('Pasien tidak terkena diabetes')
else:
      print ('Pasien terkena diabetes')
```

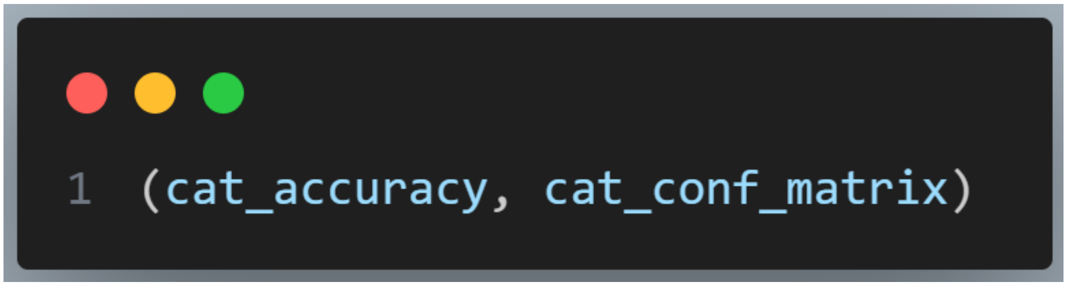
4. Evaluasi Algoritma Naïve Bayes



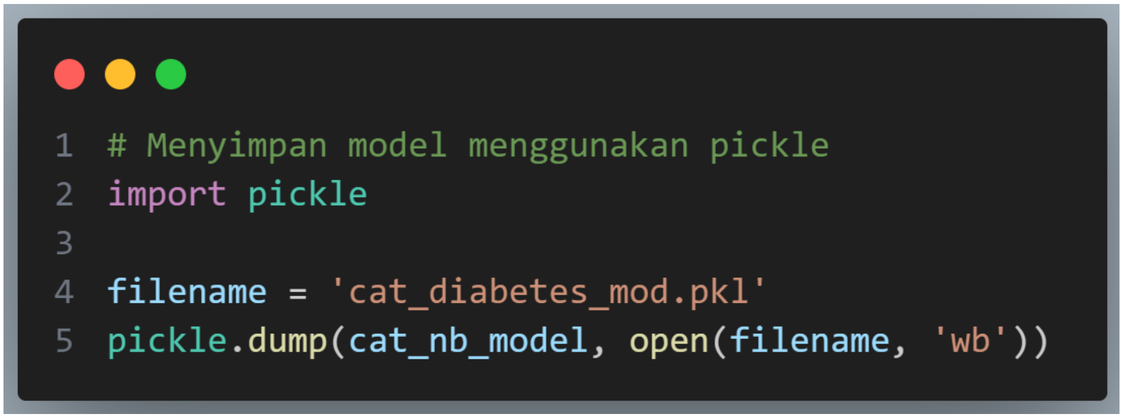
```
# Calculating accuracy for Categorical Naive Bayes
cat_accuracy = accuracy_score(y_test_cat, y_pred_cat)
```



```
# Generating the confusion matrix
cat_conf_matrix = confusion_matrix(y_test_cat, y_pred_cat)
```

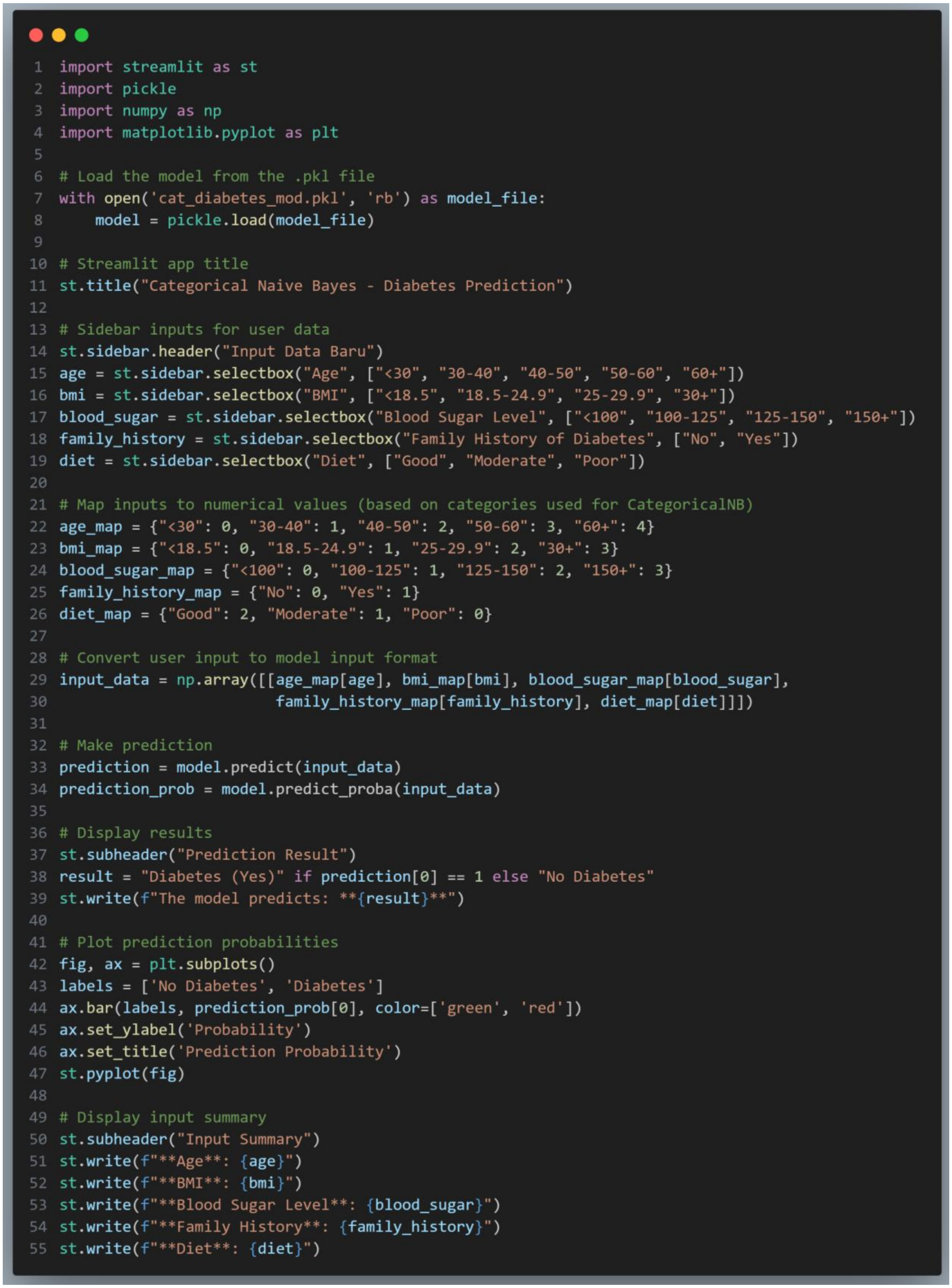


```
(cat_accuracy, cat_conf_matrix)
```



```
# Menyimpan model menggunakan pickle
import pickle

filename = 'cat_diabetes_mod.pkl'
pickle.dump(cat_nb_model, open(filename, 'wb'))
```

5. Aplikasi Menggunakan algoritma naïve bayes



```python
import streamlit as st
import pickle
import numpy as np
import matplotlib.pyplot as plt

# Load the model from the .pkl file
with open('cat_diabetes_mod.pkl', 'rb') as model_file:
    model = pickle.load(model_file)

# Streamlit app title
st.title("Categorical Naive Bayes - Diabetes Prediction")

# Sidebar inputs for user data
st.sidebar.header("Input Data Baru")
age = st.sidebar.selectbox("Age", ["<30", "30-40", "40-50", "50-60", "60+"])
bmi = st.sidebar.selectbox("BMI", ["<18.5", "18.5-24.9", "25-29.9", "30+"])
blood_sugar = st.sidebar.selectbox("Blood Sugar Level", ["<100", "100-125", "125-150", "150+"])
family_history = st.sidebar.selectbox("Family History of Diabetes", ["No", "Yes"])
diet = st.sidebar.selectbox("Diet", ["Good", "Moderate", "Poor"])

# Map inputs to numerical values (based on categories used for CategoricalNB)
age_map = {"<30": 0, "30-40": 1, "40-50": 2, "50-60": 3, "60+": 4}
bmi_map = {"<18.5": 0, "18.5-24.9": 1, "25-29.9": 2, "30+": 3}
blood_sugar_map = {"<100": 0, "100-125": 1, "125-150": 2, "150+": 3}
family_history_map = {"No": 0, "Yes": 1}
diet_map = {"Good": 2, "Moderate": 1, "Poor": 0}

# Convert user input to model input format
input_data = np.array([[age_map[age], bmi_map[bmi], blood_sugar_map[blood_sugar],
                        family_history_map[family_history], diet_map[diet]]])

# Make prediction
prediction = model.predict(input_data)
prediction_prob = model.predict_proba(input_data)

# Display results
st.subheader("Prediction Result")
result = "Diabetes (Yes)" if prediction[0] == 1 else "No Diabetes"
st.write(f"The model predicts: **{result}**")

# Plot prediction probabilities
fig, ax = plt.subplots()
labels = ['No Diabetes', 'Diabetes']
ax.bar(labels, prediction_prob[0], color=['green', 'red'])
ax.set_ylabel('Probability')
ax.set_title('Prediction Probability')
st.pyplot(fig)

# Display input summary
st.subheader("Input Summary")
st.write(f"**Age**: {age}")
st.write(f"**BMI**: {bmi}")
st.write(f"**Blood Sugar Level**: {blood_sugar}")
st.write(f"**Family History**: {family_history}")
st.write(f"**Diet**: {diet}")
```