LAMPIRAN A

Dokumentasi Kegiatan

Gambar A.1 Pertemuan dengan Wakil Kesiswaan SMKN 4 Kota Tangerang

Gambar A.2 Ruang Guru SMKN 4 Kota Tangerang

LAMPIRAN B

Surat Penelitian Skripsi

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET, DAN TEKNOLOGI
UNIVERSITAS SULTAN AGENG TIRTAYASA
FAKULTAS TEKNIK
Jalan Jendral Soedirman Km. 3 Kota Cilegon Provinsi Banten 42435
Telepon (0254) 376712 Laman : ft.untirta.ac.id

Nomor : 719 /UN.43.3.7/KT/ 2023 11 Mei 2023
Lampiran :
Hal : Permohonan Penelitian Tugas Akhir / Skripsi

Kepada Yth,
Kepala Sekolah SMKN 4 Kota Tangerang

Di
Tangerang

Sehubungan dengan rencana Penyusunan Tugas Akhir/Skripsi bagi mahasiswa kami, dengan ini mengajukan permohonan tempat penelitian di Perusahaan/Lembaga yang Bapak/Ibu pimpin.

Adapun data mahasiswa yang bersangkutan adalah sebagai berikut.

Nama	:	RAPHAEL KUSUMO HENDRIANTO
NIM	:	3332180002
Fakultas	:	TEKNIK
Jurusan/Program Studi	:	Teknik Elektro
Semester	:	Genap
Telepon / HP	:	085213170921
Durasi (Lama Penelitian)	:	2 Bulan
Rencana Topik	:	"Perbandingan Algoritma Machine Learning Dalam Memprediksi Performa Siswa"

Demikian permohonan kami sampaikan atas kerjasamanya dan perhatian Bapak/Ibu kami ucapkan terima kasih.

Wakil Dekan I,

Dr. Ir. Supriyanto, ST., M.Sc., IPM.
NIP. 197605082003121002

Tembusan :

- Ketua Program Studi Teknik Elektro

Gambar B.1 Surat Permohonan Skripsi

PEMERINTAH PROVINSI BANTEN
DINAS PENDIDIKAN DAN KEBUDAYAAN
UNIT PELAKSANA TEKNIS
SMK NEGERI 4 TANGERANG
Jl. Veteran No. 1A Telepon : (021) 5523429 Email : smkn4kotatng@yahoo.co.id

SURAT KETERANGAN
Nomor : 421.5 / 1609 - TU / 2023

Yang bertandatangan di bawah ini Kepala Sekolah Menengah Kejuruan (SMK) Negeri 4 Tangerang menerangkan bahwa :

N a m a	:	**RAFHAEL KUSUMO HENDRIANTO**
Tempat Tanggal Lahir	:	Tangerang, 24 Februari 2000
NIM	:	3332180002
Fakultas	:	Teknik
Program Studi	:	Teknik Elektro

Nama tersebut di atas adalah benar melakukan Riset/Penelitian di SMK Negeri 4 Tangerang pada bulan Mei s.d Agustus 2023, dalam rangka Penulisan Skripsi untuk menyelesaikan studi pada Universitas Sultan Ageng Tirtayasa, Dengan judul penelitian :

"PERBANDINGAN ALGORITMA MACHINE LEARNING DALAM MEMPREDIKSI PERFORMA SISWA. "

Demikian surat keterangan ini dibuat, untuk dipergunakan sebagaimana mestinya.

Tangerang, 11 September 2023
Plt. Kepala,

PEMERINTAH PROVINSI BANTEN
UNIT PELAKSANA TEKNIS
SMK NEGERI 4
TANGERANG
DINAS PENDIDIKAN DAN KEBUDAYAAN

Hj. ENDAH RESMIATI, S. Pd., M.Si
Pembina Tk 1, IV/b
NIP. 19640410 198903 2 008

Gambar B.2 Surat Keterangan Skripsi Dari SMKN 4 Kota Tangerang

LAMPIRAN C

Listing Program

1. *Decision Tree* Matematika

```
# Import Library
import numpy as np
import pandas as pd
from sklearn.metrics import accuracy_score
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier

# Membaca Data dari file Excel
df = pd.read_excel(r'''C:\Users\RAPHAEL
KUSUMO\OneDrive\Documents\Skripsi 2023 Juli\SORTER DATA
GFORM4.XLSX''')
df.head()

# Transformasi Data dari Huruf jadi angka
from sklearn.preprocessing import LabelEncoder
le_Grade = LabelEncoder()
df['SEX'] = le_Grade.fit_transform(df['SEX'])
df['UMUR'] = le_Grade.fit_transform(df['UMUR'])
df['TINGKAT'] = le_Grade.fit_transform(df['TINGKAT'])
df['JURUSAN'] = le_Grade.fit_transform(df['JURUSAN'])
df['MJOB'] = le_Grade.fit_transform(df['MJOB'])
df['FJOB'] = le_Grade.fit_transform(df['FJOB'])
df['Tingkat edukasi Ibu'] = le_Grade.fit_transform(df['Tingkat
edukasi Ibu'])
df['Tingkat edukasi Ayah'] = le_Grade.fit_transform(df['Tingkat
edukasi Ayah'])
df['EKSTRAKULIKULER'] =
le_Grade.fit_transform(df['EKSTRAKULIKULER'])
df['JURUSAN'] = le_Grade.fit_transform(df['JURUSAN'])
df['HIGHER'] = le_Grade.fit_transform(df['HIGHER'])
df['HEALTH'] = le_Grade.fit_transform(df['HEALTH'])
df['TRAVELTIME'] = le_Grade.fit_transform(df['TRAVELTIME'])
df['FAMTREE'] = le_Grade.fit_transform(df['FAMTREE'])
df['TINGGALDENGANORTU'] =
le_Grade.fit_transform(df['TINGGALDENGANORTU'])
df['REASON'] = le_Grade.fit_transform(df['REASON'])
df['BIMBEL'] = le_Grade.fit_transform(df['BIMBEL'])
df['INTERNET'] = le_Grade.fit_transform(df['INTERNET'])
df['ROMANTIC'] = le_Grade.fit_transform(df['ROMANTIC'])
df['HANGOUT'] = le_Grade.fit_transform(df['HANGOUT'])
df['ONLINEGAME'] = le_Grade.fit_transform(df['ONLINEGAME'])
df['SOSMED'] = le_Grade.fit_transform(df['SOSMED'])
df['STUDYTIME'] = le_Grade.fit_transform(df['STUDYTIME'])
df['FREETIME'] = le_Grade.fit_transform(df['FREETIME'])
df['ORGANISASI'] = le_Grade.fit_transform(df['ORGANISASI'])
df.head()

# Pembagian Target data jadi passed dan Failed
conditions = [
    (df['MATEMATIKAS3'] >= 80),
    (df['MATEMATIKAS3'] <= 79)
        ]
values = ['P', 'F']
df['Score'] = np.select(conditions, values)
df.head()
```

```
# Tampilkan DataFrame sebelum menghapus fitur
print("DataFrame Sebelum:")
print(df)
# Menghapus beberapa fitur (kolom)
fitur_yang_dihapus = ['MATEMATIKAS3','INGGRISS3', 'Nama',
'TINGKAT', 'JURUSAN','UMUR', 'SEX', 'PKNS3', 'BINDONESIAS3']
df = df.drop(columns=fitur_yang_dihapus)

# Tampilkan DataFrame setelah menghapus fitur
print("\nDataFrame Setelah:")
print(df)


#jumlah data P dan F pada target
df['Score'].value_counts()

#Membuat Variabel target
inputs = df.drop('Score',axis='columns')
target = df['Score']
inputs.head()

#Pembagian data
from sklearn.model_selection import train_test_split # Import
train_test_split function
pred_train, pred_test, tar_train, tar_test =
train_test_split(inputs, target, test_size=0.4, random_state=0)

# Menghitung sebaran data pada test set
tar_test.value_counts()

# Menghitung sebaran data pada test set
tar_train.value_counts()

# Membuat  decision tree hyperparameter
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RandomizedSearchCV
Parameters = {'max_depth' : (1,2,3,4,5,6,7,8,9),
              'criterion' : ('gini', 'entropy'),
               'max_features' : ('auto', 'sqrt', 'log2'),
              'min_samples_split' :
(2,3,4,5,6,7,8,9,10,11,12,13,14,16,18,20)
              }
DT_grid =
RandomizedSearchCV(DecisionTreeClassifier(random_state=0),
param_distributions = Parameters, cv = 10)
DT_grid.fit(pred_train,tar_train)
DT_grid.best_estimator_

from sklearn.ensemble import BaggingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Create a Decision tree classifier
gnb_classifier = DecisionTreeClassifier(max_depth=150,
```

```
max_features='log2', min_samples_split=16,
                        random_state=0)

# Create a BaggingClassifier with decision tree as the base
estimator
bagging_classifier =
BaggingClassifier(base_estimator=gnb_classifier,
n_estimators=10, random_state=0)

# Train the BaggingClassifier on the training data
bagging_classifier.fit(pred_train, tar_train)

# Make predictions on the test data
y_pred = bagging_classifier.predict(pred_test)

# Calculate accuracy
accuracy = accuracy_score(tar_test, y_pred)
print("Accuracy:", accuracy)

from sklearn.metrics import classification_report

print(classification_report(tar_test, y_pred))

from sklearn.metrics import matthews_corrcoef

mcc_score = matthews_corrcoef(tar_test, y_pred)
print("Matthews Correlation Coefficient:", mcc_score)

#Get the confusion matrix
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt

cf_matrix = confusion_matrix(tar_test, y_pred)
print(cf_matrix)

import seaborn as sns
sns.heatmap(cf_matrix, annot=True)

ax = sns.heatmap(cf_matrix, annot=True,fmt="d", cmap='Blues')

ax.set_title('Matematika DT Confusion Matrix 60:40');
ax.set_xlabel('Predictions')
ax.set_ylabel('Actuals ');

ax.xaxis.set_ticklabels(['Failed', 'Passed'])
ax.yaxis.set_ticklabels(['Failed', 'Passed'])
plt.show()
```

2. *K-Nearest Neighbour* Matematika

```
# Import Library
import numpy as np
import pandas as pd
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier

# Membaca Data dari file Excel
df = pd.read_excel(r'''C:\Users\RAPHAEL
KUSUMO\OneDrive\Documents\Skripsi 2023 Juli\SORTER DATA
GFORM4.XLSX''')
df.head()

# Transformasi Data dari Huruf jadi angka
from sklearn.preprocessing import LabelEncoder
le_Grade = LabelEncoder()
df['SEX'] = le_Grade.fit_transform(df['SEX'])
df['UMUR'] = le_Grade.fit_transform(df['UMUR'])
df['TINGKAT'] = le_Grade.fit_transform(df['TINGKAT'])
df['JURUSAN'] = le_Grade.fit_transform(df['JURUSAN'])
df['MJOB'] = le_Grade.fit_transform(df['MJOB'])
df['FJOB'] = le_Grade.fit_transform(df['FJOB'])
df['Tingkat edukasi Ibu'] = le_Grade.fit_transform(df['Tingkat
edukasi Ibu'])
df['Tingkat edukasi Ayah'] = le_Grade.fit_transform(df['Tingkat
edukasi Ayah'])
df['EKSTRAKULIKULER'] =
le_Grade.fit_transform(df['EKSTRAKULIKULER'])
df['JURUSAN'] = le_Grade.fit_transform(df['JURUSAN'])
df['HIGHER'] = le_Grade.fit_transform(df['HIGHER'])
df['HEALTH'] = le_Grade.fit_transform(df['HEALTH'])
df['TRAVELTIME'] = le_Grade.fit_transform(df['TRAVELTIME'])
df['FAMTREE'] = le_Grade.fit_transform(df['FAMTREE'])
df['TINGGALDENGANORTU'] =
le_Grade.fit_transform(df['TINGGALDENGANORTU'])
df['REASON'] = le_Grade.fit_transform(df['REASON'])
df['BIMBEL'] = le_Grade.fit_transform(df['BIMBEL'])
df['INTERNET'] = le_Grade.fit_transform(df['INTERNET'])
df['ROMANTIC'] = le_Grade.fit_transform(df['ROMANTIC'])
df['HANGOUT'] = le_Grade.fit_transform(df['HANGOUT'])
df['ONLINEGAME'] = le_Grade.fit_transform(df['ONLINEGAME'])
df['SOSMED'] = le_Grade.fit_transform(df['SOSMED'])
df['STUDYTIME'] = le_Grade.fit_transform(df['STUDYTIME'])
df['FREETIME'] = le_Grade.fit_transform(df['FREETIME'])
df['ORGANISASI'] = le_Grade.fit_transform(df['ORGANISASI'])
df.head()

# Pembagian Target data jadi passed dan Failed
conditions = [
    (df['MATEMATIKAS3'] >= 80),
    (df['MATEMATIKAS3'] <= 79)
        ]
values = ['P', 'F']
df['Score'] = np.select(conditions, values)
df.head()
```

```
# Tampilkan DataFrame sebelum menghapus fitur
print("DataFrame Sebelum:")
print(df)
# Menghapus beberapa fitur (kolom)
fitur_yang_dihapus = ['MATEMATIKAS3','INGGRISS3', 'Nama',
'TINGKAT', 'JURUSAN','UMUR', 'SEX', 'PKNS3', 'BINDONESIAS3']
df = df.drop(columns=fitur_yang_dihapus)

# Tampilkan DataFrame setelah menghapus fitur
print("\nDataFrame Setelah:")
print(df)


#jumlah data P dan F pada target
df['Score'].value_counts()

#Membuat Variabel target
inputs = df.drop('Score',axis='columns')
target = df['Score']
inputs.head()

#Pembagian data
from sklearn.model_selection import train_test_split # Import
train_test_split function
pred_train, pred_test, tar_train, tar_test =
train_test_split(inputs, target, test_size=0.4, random_state=0)

# Menghitung sebaran data pada test set
tar_test.value_counts()

# Menghitung sebaran data pada test set
tar_train.value_counts()

# finding best parameter for KNN
param_grid_knn = {
    'n_neighbors': [1, 2, 3 , 4, 5],
    'algorithm': ['ball_tree', 'kd_tree', 'brute', 'auto'],
  }
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
kNNModel_grid = GridSearchCV(estimator=KNeighborsClassifier(),
param_grid=param_grid_knn, verbose=1, cv=10, n_jobs=-1)
kNNModel_grid.fit(pred_train, tar_train)
print(kNNModel_grid.best_estimator_)

model = KNeighborsClassifier(algorithm='ball_tree',
metric='manhattan', n_neighbors=5)
classifier = model.fit(pred_train,tar_train)
classifier

predictions = classifier.predict(pred_test)
predictions

print("accuracy of training dataset
is{:.2f}".format(classifier.score(pred_train,tar_train)))
print("accuracy of test dataset is
{:.2f}".format(classifier.score(pred_test,tar_test)))
```

```
#accuracy
print("Accuracy is",accuracy_score(tar_test, predictions,
normalize = True))

from sklearn.metrics import classification_report
print(classification_report(tar_test, predictions))

#Get the confusion matrix
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt

cf_matrix = confusion_matrix(tar_test, predictions)
print(cf_matrix)

import seaborn as sns
sns.heatmap(cf_matrix, annot=True)

ax = sns.heatmap(cf_matrix, annot=True,fmt="d", cmap='Blues')

ax.set_title('Matematika KNN Confusion Matrix 60:40');
ax.set_xlabel('Predictions')
ax.set_ylabel('Actuals ');

ax.xaxis.set_ticklabels(['Failed', 'Passed'])
ax.yaxis.set_ticklabels(['Failed', 'Passed'])
plt.show()
```

3. *Support Vector Machine* Matematika

```
# Import Library
import numpy as np
import pandas as pd
from sklearn.metrics import accuracy_score
import sklearn.metrics
from sklearn import svm

# Membaca Data dari file Excel
df = pd.read_excel(r'''C:\Users\RAPHAEL
KUSUMO\OneDrive\Documents\Skripsi 2023 Juli\SORTER DATA
GFORM4.XLSX''')
df.head()

# Transformasi Data dari Huruf jadi angka
from sklearn.preprocessing import LabelEncoder
le_Grade = LabelEncoder()
df['SEX'] = le_Grade.fit_transform(df['SEX'])
df['UMUR'] = le_Grade.fit_transform(df['UMUR'])
df['TINGKAT'] = le_Grade.fit_transform(df['TINGKAT'])
df['JURUSAN'] = le_Grade.fit_transform(df['JURUSAN'])
df['MJOB'] = le_Grade.fit_transform(df['MJOB'])
df['FJOB'] = le_Grade.fit_transform(df['FJOB'])
df['Tingkat edukasi Ibu'] = le_Grade.fit_transform(df['Tingkat
edukasi Ibu'])
df['Tingkat edukasi Ayah'] = le_Grade.fit_transform(df['Tingkat
edukasi Ayah'])
```

```
df['EKSTRAKULIKULER']                                                    =
le_Grade.fit_transform(df['EKSTRAKULIKULER'])
df['JURUSAN'] = le_Grade.fit_transform(df['JURUSAN'])
df['HIGHER'] = le_Grade.fit_transform(df['HIGHER'])
df['HEALTH'] = le_Grade.fit_transform(df['HEALTH'])
df['TRAVELTIME'] = le_Grade.fit_transform(df['TRAVELTIME'])
df['FAMTREE'] = le_Grade.fit_transform(df['FAMTREE'])
df['TINGGALDENGANORTU'] =
le_Grade.fit_transform(df['TINGGALDENGANORTU'])
df['REASON'] = le_Grade.fit_transform(df['REASON'])
df['BIMBEL'] = le_Grade.fit_transform(df['BIMBEL'])
df['INTERNET'] = le_Grade.fit_transform(df['INTERNET'])
df['ROMANTIC'] = le_Grade.fit_transform(df['ROMANTIC'])
df['HANGOUT'] = le_Grade.fit_transform(df['HANGOUT'])
df['ONLINEGAME'] = le_Grade.fit_transform(df['ONLINEGAME'])
df['SOSMED'] = le_Grade.fit_transform(df['SOSMED'])
df['STUDYTIME'] = le_Grade.fit_transform(df['STUDYTIME'])
df['FREETIME'] = le_Grade.fit_transform(df['FREETIME'])
df['ORGANISASI'] = le_Grade.fit_transform(df['ORGANISASI'])
df.head()

# Pembagian Target data jadi passed dan Failed
conditions = [
    (df['MATEMATIKAS3'] >= 80),
    (df['MATEMATIKAS3'] <= 79)
        ]
values = ['P', 'F']
df['Score'] = np.select(conditions, values)
df.head()

# Tampilkan DataFrame sebelum menghapus fitur
print("DataFrame Sebelum:")
print(df)
# Menghapus beberapa fitur (kolom)
fitur_yang_dihapus = ['MATEMATIKAS3','INGGRISS3', 'Nama',
'TINGKAT', 'JURUSAN','UMUR', 'SEX', 'PKNS3', 'BINDONESIAS3']
df = df.drop(columns=fitur_yang_dihapus)

# Tampilkan DataFrame setelah menghapus fitur
print("\nDataFrame Setelah:")
print(df)

#jumlah data P dan F pada target
df['Score'].value_counts()

inputs = df.drop('Score',axis='columns')
target = df['Score']
inputs.head()

#Pembagian data
from sklearn.model_selection import train_test_split # Import
train_test_split function
pred_train, pred_test, tar_train, tar_test =
train_test_split(inputs, target, test_size=0.4, random_state=0)

# finding best parameter for SVM
param_grid = {
```

```
  'C': [0.1, 1, 10, 100, 1000],
  'gamma': [1, 0.1, 0.01, 0.001, 0.0001,1000]
    }

from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
grid = GridSearchCV(SVC(), param_grid, refit=True, verbose=3)
grid.fit(pred_train,tar_train)
grid.best_params_

model = svm.SVC(C= 10, gamma= 0.001)
classifier = model.fit(pred_train,tar_train)
classifier

predictions = classifier.predict(pred_test)
predictions

print("accuracy of training dataset
is{:.2f}".format(classifier.score(pred_train,tar_train)))
print("accuracy of test dataset is
{:.2f}".format(classifier.score(pred_test,tar_test)))
#accuracy
print("Accuracy is",accuracy_score(tar_test, predictions,
normalize = True))

predictions = classifier.predict(pred_test)
predictions

print("accuracy of training dataset
is{:.2f}".format(classifier.score(pred_train,tar_train)))
print("accuracy of test dataset is
{:.2f}".format(classifier.score(pred_test,tar_test)))
#accuracy
print("Accuracy is",accuracy_score(tar_test, predictions,
normalize = True))

from sklearn.metrics import classification_report
print(classification_report(tar_test, predictions))

#Get the confusion matrix
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt

cf_matrix = confusion_matrix(tar_test, predictions)
print(cf_matrix)

cf_matrix = confusion_matrix(tar_test, predictions)
print(cf_matrix)

import seaborn as sns
sns.heatmap(cf_matrix, annot=True)

ax = sns.heatmap(cf_matrix, annot=True,fmt="d", cmap='Blues')

ax.set_title('Matematika SVM Confusion Matrix 60:40');
ax.set_xlabel('Predictions')
ax.set_ylabel('Actuals ');
```

```
ax.xaxis.set_ticklabels(['Failed', 'Passed'])
ax.yaxis.set_ticklabels(['Failed', 'Passed'])
plt.show()
```

4. *Naive Bayes* Matematika

```
# Import Library
import numpy as np
import pandas as pd
from sklearn.metrics import accuracy_score
import sklearn.metrics
from sklearn import svm

# Membaca Data dari file Excel
df = pd.read_excel(r'''C:\Users\RAPHAEL
KUSUMO\OneDrive\Documents\Skripsi 2023 Juli\SORTER DATA
GFORM4.XLSX''')
df.head()

# Transformasi Data dari Huruf jadi angka
from sklearn.preprocessing import LabelEncoder
le_Grade = LabelEncoder()
df['SEX'] = le_Grade.fit_transform(df['SEX'])
df['UMUR'] = le_Grade.fit_transform(df['UMUR'])
df['TINGKAT'] = le_Grade.fit_transform(df['TINGKAT'])
df['JURUSAN'] = le_Grade.fit_transform(df['JURUSAN'])
df['MJOB'] = le_Grade.fit_transform(df['MJOB'])
df['FJOB'] = le_Grade.fit_transform(df['FJOB'])
df['Tingkat edukasi Ibu'] = le_Grade.fit_transform(df['Tingkat
edukasi Ibu'])
df['Tingkat edukasi Ayah'] = le_Grade.fit_transform(df['Tingkat
edukasi Ayah'])
df['EKSTRAKULIKULER'] =
le_Grade.fit_transform(df['EKSTRAKULIKULER'])
df['JURUSAN'] = le_Grade.fit_transform(df['JURUSAN'])
df['HIGHER'] = le_Grade.fit_transform(df['HIGHER'])
df['HEALTH'] = le_Grade.fit_transform(df['HEALTH'])
df['TRAVELTIME'] = le_Grade.fit_transform(df['TRAVELTIME'])
df['FAMTREE'] = le_Grade.fit_transform(df['FAMTREE'])
df['TINGGALDENGANORTU'] =
le_Grade.fit_transform(df['TINGGALDENGANORTU'])
df['REASON'] = le_Grade.fit_transform(df['REASON'])
df['BIMBEL'] = le_Grade.fit_transform(df['BIMBEL'])
df['INTERNET'] = le_Grade.fit_transform(df['INTERNET'])
df['ROMANTIC'] = le_Grade.fit_transform(df['ROMANTIC'])
df['HANGOUT'] = le_Grade.fit_transform(df['HANGOUT'])
df['ONLINEGAME'] = le_Grade.fit_transform(df['ONLINEGAME'])
df['SOSMED'] = le_Grade.fit_transform(df['SOSMED'])
df['STUDYTIME'] = le_Grade.fit_transform(df['STUDYTIME'])
df['FREETIME'] = le_Grade.fit_transform(df['FREETIME'])
df['ORGANISASI'] = le_Grade.fit_transform(df['ORGANISASI'])
df.head()
```

```
# Pembagian Target data jadi passed dan Failed
conditions = [
    (df['MATEMATIKAS3'] >= 80),
    (df['MATEMATIKAS3'] <= 79)
        ]
values = ['P', 'F']
df['Score'] = np.select(conditions, values)
df.head()

# Tampilkan DataFrame sebelum menghapus fitur
print("DataFrame Sebelum:")
print(df)
# Menghapus beberapa fitur (kolom)
fitur_yang_dihapus = ['MATEMATIKAS3','INGGRISS3', 'Nama',
'TINGKAT', 'JURUSAN','UMUR', 'SEX', 'PKNS3', 'BINDONESIAS3']
df = df.drop(columns=fitur_yang_dihapus)

# Tampilkan DataFrame setelah menghapus fitur
print("\nDataFrame Setelah:")
print(df)

#jumlah data P dan F pada target
df['Score'].value_counts()

inputs = df.drop('Score',axis='columns')
target = df['Score']
inputs.head()

#Pembagian data
from sklearn.model_selection import train_test_split # Import
train_test_split function
pred_train, pred_test, tar_train, tar_test =
train_test_split(inputs, target, test_size=0.4, random_state=0)

model = GaussianNB()
classifier = model.fit(pred_train,tar_train)
classifier

predictions = classifier.predict(pred_test)
predictions

print("accuracy of training dataset
is{:.2f}".format(classifier.score(pred_train,tar_train)))
print("accuracy of test dataset is
{:.2f}".format(classifier.score(pred_test,tar_test)))
#accuracy
print("Accuracy is",accuracy_score(tar_test, predictions,
normalize = True))

from sklearn.metrics import classification_report
print(classification_report(tar_test, predictions))

#Get the confusion matrix
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt

cf_matrix = confusion_matrix(tar_test, predictions)
print(cf_matrix)
```

```
import seaborn as sns
sns.heatmap(cf_matrix, annot=True)

ax = sns.heatmap(cf_matrix, annot=True,fmt="d", cmap='Blues')

ax.set_title('Matematika NB Confusion Matrix 60:40');
ax.set_xlabel('Predictions')
ax.set_ylabel('Actuals ');

ax.xaxis.set_ticklabels(['Failed', 'Passed'])
ax.yaxis.set_ticklabels(['Failed', 'Passed'])
plt.show()

from sklearn.metrics import matthews_corrcoef
mcc_score = matthews_corrcoef(tar_test, predictions)
print("Matthews Correlation Coefficient:", mcc_score)
```

5. *Decision Tree* Bahasa Inggris

```
import numpy as np
import pandas as pd
# To calculate the accuracy score of the model
from sklearn.metrics import accuracy_score
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier

df = pd.read_excel(r'''C:\Users\RAPHAEL
KUSUMO\OneDrive\Documents\Skripsi 2023 Juli\SORTER DATA
GFORM4.XLSX''')
df.head()

from sklearn.preprocessing import LabelEncoder
le_Grade = LabelEncoder()
df['SEX'] = le_Grade.fit_transform(df['SEX'])
df['UMUR'] = le_Grade.fit_transform(df['UMUR'])
df['TINGKAT'] = le_Grade.fit_transform(df['TINGKAT'])
df['JURUSAN'] = le_Grade.fit_transform(df['JURUSAN'])
df['MJOB'] = le_Grade.fit_transform(df['MJOB'])
df['FJOB'] = le_Grade.fit_transform(df['FJOB'])
df['Tingkat edukasi Ibu'] = le_Grade.fit_transform(df['Tingkat
edukasi Ibu'])
df['Tingkat edukasi Ayah'] = le_Grade.fit_transform(df['Tingkat
edukasi Ayah'])
df['EKSTRAKULIKULER'] =
le_Grade.fit_transform(df['EKSTRAKULIKULER'])
df['JURUSAN'] = le_Grade.fit_transform(df['JURUSAN'])
df['HIGHER'] = le_Grade.fit_transform(df['HIGHER'])
df['HEALTH'] = le_Grade.fit_transform(df['HEALTH'])
df['TRAVELTIME'] = le_Grade.fit_transform(df['TRAVELTIME'])
df['FAMTREE'] = le_Grade.fit_transform(df['FAMTREE'])
df['TINGGALDENGANORTU'] =
le_Grade.fit_transform(df['TINGGALDENGANORTU'])
df['REASON'] = le_Grade.fit_transform(df['REASON'])
df['BIMBEL'] = le_Grade.fit_transform(df['BIMBEL'])
```

```
df['INTERNET'] = le_Grade.fit_transform(df['INTERNET'])
df['ROMANTIC'] = le_Grade.fit_transform(df['ROMANTIC'])
df['HANGOUT'] = le_Grade.fit_transform(df['HANGOUT'])
df['ONLINEGAME'] = le_Grade.fit_transform(df['ONLINEGAME'])
df['SOSMED'] = le_Grade.fit_transform(df['SOSMED'])
df['STUDYTIME'] = le_Grade.fit_transform(df['STUDYTIME'])
df['FREETIME'] = le_Grade.fit_transform(df['FREETIME'])
df['ORGANISASI'] = le_Grade.fit_transform(df['ORGANISASI'])
df.head()

conditions = [
    (df['INGGRISS3'] >= 80),
    (df['INGGRISS3']<= 79)
        ]
values = ['P', 'F']
df['Score'] = np.select(conditions, values)
df.head(5)

# Tampilkan DataFrame sebelum menghapus fitur
print("DataFrame Sebelum:")
print(df)

# Menghapus beberapa fitur (kolom)
fitur_yang_dihapus = ['MATEMATIKAS3','INGGRISS3', 'Nama',
'TINGKAT', 'JURUSAN','UMUR', 'SEX', 'PKNS3', 'BINDONESIAS3']
df = df.drop(columns=fitur_yang_dihapus)

# Tampilkan DataFrame setelah menghapus fitur
print("\nDataFrame Setelah:")
print(df)

#jumlah data 1 dan 0 pada target
df['Score'].value_counts()

inputs = df.drop('Score',axis='columns')
target = df['Score']
inputs.head()

#Pembagian data
from sklearn.model_selection import train_test_split # Import
train_test_split function
pred_train, pred_test, tar_train, tar_test =
train_test_split(inputs, target, test_size=0.4, random_state=0)

#Decision Tree Hyperparameter
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RandomizedSearchCV
Parameters = {'max_depth' : (1,2,3,4,5,6,7,8,9),
              'criterion' : ('gini', 'entropy'),
               'max_features' : ('auto', 'sqrt', 'log2'),
              'min_samples_split' :
(2,3,4,5,6,7,8,9,10,11,12,13,14,16,18,20)
              }
DT_grid =
RandomizedSearchCV(DecisionTreeClassifier(random_state=0),
param_distributions = Parameters, cv = 10)
DT_grid.fit(pred_train,tar_train)
DT_grid.best_estimator_
```

```
from sklearn.ensemble import BaggingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Create a Decision Tree classifier
gnb_classifier = DecisionTreeClassifier(max_depth=150,
max_features='log2', min_samples_split=16,
                        random_state=0)

# Create a BaggingClassifier with Decision Tree as the base
estimator
bagging_classifier =
BaggingClassifier(base_estimator=gnb_classifier,
n_estimators=10, random_state=0)

# Train the BaggingClassifier on the training data
bagging_classifier.fit(pred_train, tar_train)

# Make predictions on the test data
y_pred = bagging_classifier.predict(pred_test)

# Calculate accuracy
accuracy = accuracy_score(tar_test, y_pred)
print("Accuracy:", accuracy)

from sklearn.metrics import classification_report
print(classification_report(tar_test, y_pred))

from sklearn.metrics import matthews_corrcoef
mcc_score = matthews_corrcoef(tar_test, y_pred)
print("Matthews Correlation Coefficient:", mcc_score)

#Get the confusion matrix
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt

cf_matrix = confusion_matrix(tar_test, y_pred)
print(cf_matrix)

import seaborn as sns
sns.heatmap(cf_matrix, annot=True)

ax = sns.heatmap(cf_matrix, annot=True,fmt="d", cmap='Blues')

ax.set_title('Matematika DT Confusion Matrix 60:40');
ax.set_xlabel('Predictions')
ax.set_ylabel('Actuals ');

ax.xaxis.set_ticklabels(['Failed', 'Passed'])
ax.yaxis.set_ticklabels(['Failed', 'Passed'])
plt.show()

from sklearn.tree import export_graphviz
import os
```

```
# Create a directory to store the DOT files
output_directory = "tree_visualizations"
os.makedirs(output_directory, exist_ok=True)

# Access and visualize individual trees
for i, tree in enumerate(bagging_classifier.estimators_):
    # Define the output file path and format
    output_file = os.path.join(output_directory,
f'tree_{i}.pdf')

    # Export the tree to the specified format and file
    export_graphviz(tree, out_file=output_file,
feature_names=inputs.columns, class_names=['Passed', 'Failed'],
filled=True, rounded=True)
```

6. *K-Nearest Neighbour* Bahasa Inggris

```
import numpy as np
import pandas as pd
# To calculate the accuracy score of the model
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier

df = pd.read_excel(r'''C:\Users\RAPHAEL
KUSUMO\OneDrive\Documents\Skripsi 2023 Juli\SORTER DATA
GFORM4.XLSX''')
df.head()


from sklearn.preprocessing import LabelEncoder
le_Grade = LabelEncoder()
from sklearn.preprocessing import LabelEncoder
le_Grade = LabelEncoder()
df['SEX'] = le_Grade.fit_transform(df['SEX'])
df['UMUR'] = le_Grade.fit_transform(df['UMUR'])
df['TINGKAT'] = le_Grade.fit_transform(df['TINGKAT'])
df['JURUSAN'] = le_Grade.fit_transform(df['JURUSAN'])
df['MJOB'] = le_Grade.fit_transform(df['MJOB'])
df['FJOB'] = le_Grade.fit_transform(df['FJOB'])
df['Tingkat edukasi Ibu'] = le_Grade.fit_transform(df['Tingkat
edukasi Ibu'])
df['Tingkat edukasi Ayah'] = le_Grade.fit_transform(df['Tingkat
edukasi Ayah'])
df['EKSTRAKULIKULER'] =
le_Grade.fit_transform(df['EKSTRAKULIKULER'])
df['JURUSAN'] = le_Grade.fit_transform(df['JURUSAN'])
df['HIGHER'] = le_Grade.fit_transform(df['HIGHER'])
df['HEALTH'] = le_Grade.fit_transform(df['HEALTH'])
df['TRAVELTIME'] = le_Grade.fit_transform(df['TRAVELTIME'])
df['FAMTREE'] = le_Grade.fit_transform(df['FAMTREE'])
df['TINGGALDENGANORTU'] =
le_Grade.fit_transform(df['TINGGALDENGANORTU'])
df['REASON'] = le_Grade.fit_transform(df['REASON'])
df['BIMBEL'] = le_Grade.fit_transform(df['BIMBEL'])
df['INTERNET'] = le_Grade.fit_transform(df['INTERNET'])
df['ROMANTIC'] = le_Grade.fit_transform(df['ROMANTIC'])
```

```
df['HANGOUT'] = le_Grade.fit_transform(df['HANGOUT'])
df['ONLINEGAME'] = le_Grade.fit_transform(df['ONLINEGAME'])
df['SOSMED'] = le_Grade.fit_transform(df['SOSMED'])
df['STUDYTIME'] = le_Grade.fit_transform(df['STUDYTIME'])
df['FREETIME'] = le_Grade.fit_transform(df['FREETIME'])
df['ORGANISASI'] = le_Grade.fit_transform(df['ORGANISASI'])
df.head()

conditions = [
    (df['INGGRISS3'] >= 80),
    (df['INGGRISS3']<= 79)
        ]
values = ['P', 'F']
df['Score'] = np.select(conditions, values)
df.head(5)

# Tampilkan DataFrame sebelum menghapus fitur
print("DataFrame Sebelum:")
print(df)

# Menghapus beberapa fitur (kolom)
fitur_yang_dihapus = ['MATEMATIKAS3','INGGRISS3', 'Nama',
'TINGKAT', 'JURUSAN','UMUR', 'SEX', 'PKNS3', 'BINDONESIAS3']
df = df.drop(columns=fitur_yang_dihapus)

# Tampilkan DataFrame setelah menghapus fitur
print("\nDataFrame Setelah:")
print(df)

#jumlah data 1 dan 0 pada target
df['Score'].value_counts()

inputs = df.drop('Score',axis='columns')
target = df['Score']
inputs.head()

#Pembagian data
from sklearn.model_selection import train_test_split # Import
train_test_split function
pred_train, pred_test, tar_train, tar_test =
train_test_split(inputs, target, test_size=0.4, random_state=0)

# finding best parameter for KNN
param_grid_knn = {
    'n_neighbors': [1, 2, 3 , 4, 5,6, 7, 8, 9, 10, 11, 12, 13,
14, 15 ,16 ,17 ,18,19, 20],
    'algorithm': ['ball_tree', 'kd_tree', 'brute', 'auto'],
    'metric': ['minkowski', 'euclidean', 'manhattan',
'chebyshev']
}

from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
kNNModel_grid = GridSearchCV(estimator=KNeighborsClassifier(),
param_grid=param_grid_knn, verbose=1, cv=10, n_jobs=-1)
kNNModel_grid.fit(pred_train, tar_train)
print(kNNModel_grid.best_estimator_)
```

```
model = KNeighborsClassifier(algorithm='ball_tree',
n_neighbors=6)
classifier = model.fit(pred_train,tar_train)
classifier

predictions = classifier.predict(pred_test)
predictions

print("accuracy of training dataset
is{:.2f}".format(classifier.score(pred_train,tar_train)))
print("accuracy of test dataset is
{:.2f}".format(classifier.score(pred_test,tar_test)))
#accuracy
print("Accuracy is",accuracy_score(tar_test, predictions,
normalize = True))

from sklearn.metrics import classification_report
print(classification_report(tar_test, predictions))

#Get the confusion matrix
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt

cf_matrix = confusion_matrix(tar_test, predictions)
print(cf_matrix)

import seaborn as sns
sns.heatmap(cf_matrix, annot=True)

ax = sns.heatmap(cf_matrix, annot=True,fmt="d", cmap='Blues')

ax.set_title('Bahasa Inggris KNN Confusion Matrix 60:40');
ax.set_xlabel('Predictions')
ax.set_ylabel('Actuals ');

ax.xaxis.set_ticklabels(['Failed', 'Passed'])
ax.yaxis.set_ticklabels(['Failed', 'Passed'])
plt.show()

from sklearn.metrics import matthews_corrcoef
mcc_score = matthews_corrcoef(tar_test, predictions)
print("Matthews Correlation Coefficient:", mcc_score)
```

7. *Support Vector Machine* Bahasa Inggris

```
import numpy as np
import pandas as pd
# To calculate the accuracy score of the model
from sklearn.metrics import accuracy_score
import sklearn.metrics
from sklearn import svm
```

```
df = pd.read_excel(r'''C:\Users\RAPHAEL
KUSUMO\OneDrive\Documents\Skripsi 2023 Juli\SORTER DATA
GFORM4.XLSX''')
df.head()

from sklearn.preprocessing import LabelEncoder
le_Grade = LabelEncoder()
df['SEX'] = le_Grade.fit_transform(df['SEX'])
df['UMUR'] = le_Grade.fit_transform(df['UMUR'])
df['TINGKAT'] = le_Grade.fit_transform(df['TINGKAT'])
df['JURUSAN'] = le_Grade.fit_transform(df['JURUSAN'])
df['MJOB'] = le_Grade.fit_transform(df['MJOB'])
df['FJOB'] = le_Grade.fit_transform(df['FJOB'])
df['Tingkat edukasi Ibu'] = le_Grade.fit_transform(df['Tingkat
edukasi Ibu'])
df['Tingkat edukasi Ayah'] = le_Grade.fit_transform(df['Tingkat
edukasi Ayah'])
df['EKSTRAKULIKULER'] =
le_Grade.fit_transform(df['EKSTRAKULIKULER'])
df['JURUSAN'] = le_Grade.fit_transform(df['JURUSAN'])
df['HIGHER'] = le_Grade.fit_transform(df['HIGHER'])
df['HEALTH'] = le_Grade.fit_transform(df['HEALTH'])
df['TRAVELTIME'] = le_Grade.fit_transform(df['TRAVELTIME'])
df['FAMTREE'] = le_Grade.fit_transform(df['FAMTREE'])
df['TINGGALDENGANORTU'] =
le_Grade.fit_transform(df['TINGGALDENGANORTU'])
df['REASON'] = le_Grade.fit_transform(df['REASON'])
df['BIMBEL'] = le_Grade.fit_transform(df['BIMBEL'])
df['INTERNET'] = le_Grade.fit_transform(df['INTERNET'])
df['ROMANTIC'] = le_Grade.fit_transform(df['ROMANTIC'])
df['HANGOUT'] = le_Grade.fit_transform(df['HANGOUT'])
df['ONLINEGAME'] = le_Grade.fit_transform(df['ONLINEGAME'])
df['SOSMED'] = le_Grade.fit_transform(df['SOSMED'])
df['STUDYTIME'] = le_Grade.fit_transform(df['STUDYTIME'])
df['FREETIME'] = le_Grade.fit_transform(df['FREETIME'])
df['ORGANISASI'] = le_Grade.fit_transform(df['ORGANISASI'])
df.head()

conditions = [
    (df['INGGRISS3'] >= 80),
    (df['INGGRISS3']<= 79)
        ]
values = ['1', '0']
df['Score'] = np.select(conditions, values)
df.head(5)

# Tampilkan DataFrame sebelum menghapus fitur
print("DataFrame Sebelum:")
print(df)

# Menghapus beberapa fitur (kolom)
fitur_yang_dihapus = ['MATEMATIKAS3','INGGRISS3', 'Nama',
'TINGKAT', 'JURUSAN','UMUR', 'SEX', 'PKNS3', 'BINDONESIAS3']
df = df.drop(columns=fitur_yang_dihapus)

# Tampilkan DataFrame setelah menghapus fitur
print("\nDataFrame Setelah:")
print(df)
```

```
#jumlah data 1 dan 0 pada target
df['Score'].value_counts()

inputs = df.drop('Score',axis='columns')
target = df['Score']
inputs.head()

#Pembagian data
from sklearn.model_selection import train_test_split # Import
train_test_split function
pred_train, pred_test, tar_train, tar_test =
train_test_split(inputs, target, test_size=0.4, random_state=0)

# finding best parameter for SVM
weights = {1:4.0, 0:1.0}
param_grid = {
  'C': [0.1, 1, 10, 100, 1000],
  'gamma': [1, 0.1, 0.01, 0.001, 0.0001]
 }
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
grid = GridSearchCV(svm.SVC(kernel='rbf',
class_weight=weights), param_grid, refit=True, verbose=3)
grid.fit(pred_train,tar_train)
grid.best_params_

#COba2class_weight= 'balanced'
# fit the model and get the separating hyperplane
tar_train = [int(label) for label in tar_train]
weights = {1:4.0, 0:1.0}
model = svm.SVC(kernel='rbf',C=0.1,gamma=0.01,
class_weight=weights)
classifier = model.fit(pred_train,tar_train)
classifier

tar_test = [int(label) for label in tar_test]
predictions = classifier.predict(pred_test)
predictions

predictions = classifier.predict(pred_test)
predictions

#Confusion Matrix
from sklearn import metrics
print(metrics.accuracy_score(tar_test, predictions))

print("accuracy of training dataset
is{:.2f}".format(classifier.score(pred_train,tar_train)))
print("accuracy of test dataset is
{:.2f}".format(classifier.score(pred_test,tar_test)))
#accuracy
print("Accuracy is",accuracy_score(tar_test, predictions,
normalize = True))

from sklearn.metrics import classification_report
print(classification_report(tar_test, predictions))

#Get the confusion matrix
```

```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt

cf_matrix = confusion_matrix(tar_test, predictions)
print(cf_matrix)

import seaborn as sns
sns.heatmap(cf_matrix, annot=True)

ax = sns.heatmap(cf_matrix, annot=True,fmt="d", cmap='Blues')

ax.set_title('Bahasa Inggris SVM Confusion Matrix 60:40');
ax.set_xlabel('Predictions')
ax.set_ylabel('Actuals ');

ax.xaxis.set_ticklabels(['Failed', 'Passed'])
ax.yaxis.set_ticklabels(['Failed', 'Passed'])
#ax.xaxis.set_ticklabels(['Passed', 'Failed'])
#ax.yaxis.set_ticklabels(['Passed', 'Failed'])

plt.show()

from sklearn.metrics import matthews_corrcoef
mcc_score = matthews_corrcoef(tar_test, predictions)
print("Matthews Correlation Coefficient:", mcc_score)
```

8. *Naive Bayes* Bahasa Inggris

```
import numpy as np
import pandas as pd
# To calculate the accuracy score of the model
from sklearn.metrics import accuracy_score
from sklearn.naive_bayes import GaussianNB

df = pd.read_excel(r'''C:\Users\RAPHAEL
KUSUMO\OneDrive\Documents\Skripsi 2023 Juli\SORTER DATA
GFORM4.XLSX''')
df.head()

from sklearn.preprocessing import LabelEncoder
le_Grade = LabelEncoder()
df['SEX'] = le_Grade.fit_transform(df['SEX'])
df['UMUR'] = le_Grade.fit_transform(df['UMUR'])
df['TINGKAT'] = le_Grade.fit_transform(df['TINGKAT'])
df['JURUSAN'] = le_Grade.fit_transform(df['JURUSAN'])
df['MJOB'] = le_Grade.fit_transform(df['MJOB'])
df['FJOB'] = le_Grade.fit_transform(df['FJOB'])
df['Tingkat edukasi Ibu'] = le_Grade.fit_transform(df['Tingkat
edukasi Ibu'])
df['Tingkat edukasi Ayah'] = le_Grade.fit_transform(df['Tingkat
edukasi Ayah'])
df['EKSTRAKULIKULER'] =
le_Grade.fit_transform(df['EKSTRAKULIKULER'])
df['JURUSAN'] = le_Grade.fit_transform(df['JURUSAN'])
df['HIGHER'] = le_Grade.fit_transform(df['HIGHER'])
```

```
df['HEALTH'] = le_Grade.fit_transform(df['HEALTH'])
df['TRAVELTIME'] = le_Grade.fit_transform(df['TRAVELTIME'])
df['FAMTREE'] = le_Grade.fit_transform(df['FAMTREE'])
df['TINGGALDENGANORTU'] =
le_Grade.fit_transform(df['TINGGALDENGANORTU'])
df['REASON'] = le_Grade.fit_transform(df['REASON'])
df['BIMBEL'] = le_Grade.fit_transform(df['BIMBEL'])
df['INTERNET'] = le_Grade.fit_transform(df['INTERNET'])
df['ROMANTIC'] = le_Grade.fit_transform(df['ROMANTIC'])
df['HANGOUT'] = le_Grade.fit_transform(df['HANGOUT'])
df['ONLINEGAME'] = le_Grade.fit_transform(df['ONLINEGAME'])
df['SOSMED'] = le_Grade.fit_transform(df['SOSMED'])
df['STUDYTIME'] = le_Grade.fit_transform(df['STUDYTIME'])
df['FREETIME'] = le_Grade.fit_transform(df['FREETIME'])
df['ORGANISASI'] = le_Grade.fit_transform(df['ORGANISASI'])
df.head()

conditions = [
    (df['INGGRISS3'] >= 80),
    (df['INGGRISS3']<= 79)
        ]
values = ['P', 'F']
df['Score'] = np.select(conditions, values)
df.head(5)

# Tampilkan DataFrame sebelum menghapus fitur
print("DataFrame Sebelum:")
print(df)

# Menghapus beberapa fitur (kolom)
fitur_yang_dihapus = ['MATEMATIKAS3','INGGRISS3', 'Nama',
'TINGKAT', 'JURUSAN','UMUR', 'SEX', 'PKNS3', 'BINDONESIAS3']
df = df.drop(columns=fitur_yang_dihapus)

# Tampilkan DataFrame setelah menghapus fitur
print("\nDataFrame Setelah:")
print(df)

#jumlah data 1 dan 0 pada target
df['Score'].value_counts()

inputs = df.drop('Score',axis='columns')
target = df['Score']
inputs.head()

#Pembagian data
from sklearn.model_selection import train_test_split # Import
train_test_split function
pred_train, pred_test, tar_train, tar_test =
train_test_split(inputs, target, test_size=0.4, random_state=0)


param_grid_nb = {
    'var_smoothing': np.logspace(1,-9, num=100)
}
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import GridSearchCV
```

```
nbModel_grid = GridSearchCV(estimator=GaussianNB(),
param_grid=param_grid_nb, verbose=1, cv=10, n_jobs=-1)
nbModel_grid.fit(pred_train, tar_train)
print(nbModel_grid.best_estimator_)

#Training data 60%
model = GaussianNB(var_smoothing=1.2618568830660185e-09)
classifier = model.fit(pred_train,tar_train)
classifier

predictions = classifier.predict(pred_test)
predictions

print("accuracy of training dataset
is{:.2f}".format(classifier.score(pred_train,tar_train)))
print("accuracy of test dataset is
{:.2f}".format(classifier.score(pred_test,tar_test)))
#accuracy
print("Accuracy is",accuracy_score(tar_test, predictions,
normalize = True))

from sklearn.metrics import classification_report
print(classification_report(tar_test, predictions))

#Get the confusion matrix
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt

cf_matrix = confusion_matrix(tar_test, predictions)
print(cf_matrix)

import seaborn as sns
sns.heatmap(cf_matrix, annot=True)

ax = sns.heatmap(cf_matrix, annot=True,fmt="d", cmap='Blues')

ax.set_title('Bahasa Inggris NB Confusion Matrix 60:40');
ax.set_xlabel('Predictions')
ax.set_ylabel('Actuals ');

ax.xaxis.set_ticklabels(['Failed', 'Passed'])
ax.yaxis.set_ticklabels(['Failed', 'Passed'])
plt.show()

from sklearn.metrics import matthews_corrcoef
mcc_score = matthews_corrcoef(tar_test, predictions)
print("Matthews Correlation Coefficient:", mcc_score)
```

9. *Decision Tree* PKn

```
# Import Library
import numpy as np
import pandas as pd
from sklearn.metrics import accuracy_score
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier

# Membaca Data dari file Excel
df = pd.read_excel(r'''C:\Users\RAPHAEL
KUSUMO\OneDrive\Documents\Skripsi 2023 Juli\SORTER DATA
GFORM4.XLSX''')
df.head()

# Transformasi Data dari Huruf jadi angka
from sklearn.preprocessing import LabelEncoder
le_Grade = LabelEncoder()
df['SEX'] = le_Grade.fit_transform(df['SEX'])
df['UMUR'] = le_Grade.fit_transform(df['UMUR'])
df['TINGKAT'] = le_Grade.fit_transform(df['TINGKAT'])
df['JURUSAN'] = le_Grade.fit_transform(df['JURUSAN'])
df['MJOB'] = le_Grade.fit_transform(df['MJOB'])
df['FJOB'] = le_Grade.fit_transform(df['FJOB'])
df['Tingkat edukasi Ibu'] = le_Grade.fit_transform(df['Tingkat
edukasi Ibu'])
df['Tingkat edukasi Ayah'] = le_Grade.fit_transform(df['Tingkat
edukasi Ayah'])
df['EKSTRAKULIKULER'] =
le_Grade.fit_transform(df['EKSTRAKULIKULER'])
df['JURUSAN'] = le_Grade.fit_transform(df['JURUSAN'])
df['HIGHER'] = le_Grade.fit_transform(df['HIGHER'])
df['HEALTH'] = le_Grade.fit_transform(df['HEALTH'])
df['TRAVELTIME'] = le_Grade.fit_transform(df['TRAVELTIME'])
df['FAMTREE'] = le_Grade.fit_transform(df['FAMTREE'])
df['TINGGALDENGANORTU'] =
le_Grade.fit_transform(df['TINGGALDENGANORTU'])
df['REASON'] = le_Grade.fit_transform(df['REASON'])
df['BIMBEL'] = le_Grade.fit_transform(df['BIMBEL'])
df['INTERNET'] = le_Grade.fit_transform(df['INTERNET'])
df['ROMANTIC'] = le_Grade.fit_transform(df['ROMANTIC'])
df['HANGOUT'] = le_Grade.fit_transform(df['HANGOUT'])
df['ONLINEGAME'] = le_Grade.fit_transform(df['ONLINEGAME'])
df['SOSMED'] = le_Grade.fit_transform(df['SOSMED'])
df['STUDYTIME'] = le_Grade.fit_transform(df['STUDYTIME'])
df['FREETIME'] = le_Grade.fit_transform(df['FREETIME'])
df['ORGANISASI'] = le_Grade.fit_transform(df['ORGANISASI'])
df.head()

conditions = [
    (df['PKNS3'] >= 80),
    (df['PKNS3']<= 79)
        ]
values = ['P', 'F']
df['Score'] = np.select(conditions, values)
df.head(5)
```

```
# Tampilkan DataFrame sebelum menghapus fitur
print("DataFrame Sebelum:")
print(df)

# Menghapus beberapa fitur (kolom)
fitur_yang_dihapus = ['MATEMATIKAS3','INGGRISS3', 'Nama',
'TINGKAT', 'JURUSAN','UMUR', 'SEX', 'PKNS3', 'BINDONESIAS3']
df = df.drop(columns=fitur_yang_dihapus)

# Tampilkan DataFrame setelah menghapus fitur
print("\nDataFrame Setelah:")
print(df)

#jumlah data 1 dan 0 pada target
df['Score'].value_counts()

inputs = df.drop('Score',axis='columns')
target = df['Score']
inputs.head()

#Pembagian data
from sklearn.model_selection import train_test_split # Import
train_test_split function
pred_train, pred_test, tar_train, tar_test =
train_test_split(inputs, target, test_size=0.4, random_state=0)

from sklearn.ensemble import BaggingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Create a Gaussian Naive Bayes classifier
gnb_classifier = DecisionTreeClassifier(max_depth=100,
max_features='sqrt', min_samples_split=18,
                      random_state=42)

# Create a BaggingClassifier with Gaussian Naive Bayes as the
base estimator
bagging_classifier =
BaggingClassifier(base_estimator=gnb_classifier,
n_estimators=10, random_state=0)

# Train the BaggingClassifier on the training data
bagging_classifier.fit(pred_train, tar_train)

# Make predictions on the test data
y_pred = bagging_classifier.predict(pred_test)

# Calculate accuracy
accuracy = accuracy_score(tar_test, y_pred)
print("Accuracy:", accuracy)

from sklearn.metrics import classification_report

print(classification_report(tar_test, y_pred))

from sklearn.metrics import matthews_corrcoef
```

```
from sklearn.metrics import matthews_corrcoef
mcc_score = matthews_corrcoef(tar_test, y_pred)
print("Matthews Correlation Coefficient:", mcc_score)

#Get the confusion matrix
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt

cf_matrix = confusion_matrix(tar_test, y_pred)
print(cf_matrix)

import seaborn as sns
sns.heatmap(cf_matrix, annot=True)

ax = sns.heatmap(cf_matrix, annot=True,fmt="d", cmap='Blues')

ax.set_title('Matematika DT Confusion Matrix 60:40');
ax.set_xlabel('Predictions')
ax.set_ylabel('Actuals ');

ax.xaxis.set_ticklabels(['Failed', 'Passed'])
ax.yaxis.set_ticklabels(['Failed', 'Passed'])
plt.show()

from sklearn.tree import export_graphviz
# Create a BaggingClassifier with DecisionTreeClassifier as the
base estimator
bagging_classifier =
BaggingClassifier(base_estimator=DecisionTreeClassifier(),
n_estimators=10, random_state=42)

# Train the BaggingClassifier
bagging_classifier.fit(pred_train, tar_train)

# Access and visualize individual trees
for i, tree in enumerate(bagging_classifier.estimators_):
    # Export the tree to a .dot file (replace "tree_i.dot" with
a suitable filename)
    export_graphviz(tree, out_file=f'tree_{i}.dot',
feature_names=inputs.columns, class_names=['Passed', 'Failed'],
filled=True, rounded=True)
```

10. *K-Nearest Neighbour* PKn

```
import numpy as np
import pandas as pd
# To calculate the accuracy score of the model
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier

df = pd.read_excel(r'''C:\Users\RAPHAEL
KUSUMO\OneDrive\Documents\Skripsi 2023 Juli\SORTER DATA
GFORM4.XLSX''')
df.head()
```

```
from sklearn.preprocessing import LabelEncoder
le_Grade = LabelEncoder()
df['SEX'] = le_Grade.fit_transform(df['SEX'])
df['UMUR'] = le_Grade.fit_transform(df['UMUR'])
df['TINGKAT'] = le_Grade.fit_transform(df['TINGKAT'])
df['JURUSAN'] = le_Grade.fit_transform(df['JURUSAN'])
df['MJOB'] = le_Grade.fit_transform(df['MJOB'])
df['FJOB'] = le_Grade.fit_transform(df['FJOB'])
df['Tingkat edukasi Ibu'] = le_Grade.fit_transform(df['Tingkat
edukasi Ibu'])
df['Tingkat edukasi Ayah'] = le_Grade.fit_transform(df['Tingkat
edukasi Ayah'])
df['EKSTRAKULIKULER'] =
le_Grade.fit_transform(df['EKSTRAKULIKULER'])
df['JURUSAN'] = le_Grade.fit_transform(df['JURUSAN'])
df['HIGHER'] = le_Grade.fit_transform(df['HIGHER'])
df['HEALTH'] = le_Grade.fit_transform(df['HEALTH'])
df['TRAVELTIME'] = le_Grade.fit_transform(df['TRAVELTIME'])
df['FAMTREE'] = le_Grade.fit_transform(df['FAMTREE'])
df['TINGGALDENGANORTU'] =
le_Grade.fit_transform(df['TINGGALDENGANORTU'])
df['REASON'] = le_Grade.fit_transform(df['REASON'])
df['BIMBEL'] = le_Grade.fit_transform(df['BIMBEL'])
df['INTERNET'] = le_Grade.fit_transform(df['INTERNET'])
df['ROMANTIC'] = le_Grade.fit_transform(df['ROMANTIC'])
df['HANGOUT'] = le_Grade.fit_transform(df['HANGOUT'])
df['ONLINEGAME'] = le_Grade.fit_transform(df['ONLINEGAME'])
df['SOSMED'] = le_Grade.fit_transform(df['SOSMED'])
df['STUDYTIME'] = le_Grade.fit_transform(df['STUDYTIME'])
df['FREETIME'] = le_Grade.fit_transform(df['FREETIME'])
df['ORGANISASI'] = le_Grade.fit_transform(df['ORGANISASI'])
df.head()

conditions = [
    (df['PKNS3'] >= 80),
    (df['PKNS3']<= 79)
        ]
values = ['P', 'F']
df['Score'] = np.select(conditions, values)
df.head(5)

# Tampilkan DataFrame sebelum menghapus fitur
print("DataFrame Sebelum:")
print(df)

# Menghapus beberapa fitur (kolom)
fitur_yang_dihapus = ['MATEMATIKAS3','INGGRISS3', 'Nama',
'TINGKAT', 'JURUSAN','UMUR', 'SEX', 'PKNS3', 'BINDONESIAS3']
df = df.drop(columns=fitur_yang_dihapus)

# Tampilkan DataFrame setelah menghapus fitur
print("\nDataFrame Setelah:")
print(df)

inputs = df.drop('Score',axis='columns')
target = df['Score']
inputs.head()
```

```
#Pembagian data
from sklearn.model_selection import train_test_split # Import
train_test_split function
pred_train, pred_test, tar_train, tar_test =
train_test_split(inputs, target, test_size=0.4, random_state=0)

# finding best parameter for KNN
param_grid_knn = {
    'n_neighbors': [1, 2, 3, 4, 5 , 5, 7, 8,9,10,11,12],
    'algorithm': ['ball_tree', 'kd_tree', 'brute', 'auto'],
    'metric': ['minkowski', 'euclidean', 'manhattan',
'chebyshev'],
    'weights': ['uniform', 'distance']
}

from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
kNNModel_grid = GridSearchCV(estimator=KNeighborsClassifier(),
param_grid=param_grid_knn, verbose=1, cv=5, n_jobs=-1)
kNNModel_grid.fit(pred_train, tar_train)
print(kNNModel_grid.best_estimator_)

model = KNeighborsClassifier(algorithm='ball_tree',
weights='distance')
classifier = model.fit(pred_train,tar_train)
classifier

predictions = classifier.predict(pred_test)
predictions

from sklearn.metrics import classification_report
print(classification_report(tar_test, predictions))

#Get the confusion matrix
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt

cf_matrix = confusion_matrix(tar_test, predictions)
print(cf_matrix)

import seaborn as sns
sns.heatmap(cf_matrix, annot=True)

ax = sns.heatmap(cf_matrix, annot=True,fmt="d", cmap='Blues')

ax.set_title('PKN KNN Confusion Matrix 60:40');
ax.set_xlabel('Predictions')
ax.set_ylabel('Actuals ');

ax.xaxis.set_ticklabels(['Failed', 'Passed'])
ax.yaxis.set_ticklabels(['Failed', 'Passed'])
plt.show()

from sklearn.metrics import matthews_corrcoef
mcc_score = matthews_corrcoef(tar_test, predictions)
print("Matthews Correlation Coefficient:", mcc_score)
```

```
import numpy as np
import pandas as pd
# To calculate the accuracy score of the model
from sklearn.metrics import accuracy_score
import sklearn.metrics
from sklearn import svm

df = pd.read_excel(r'''C:\Users\RAPHAEL
KUSUMO\OneDrive\Documents\Skripsi 2023 Juli\SORTER DATA
GFORM4.XLSX''')
df.head()

from sklearn.preprocessing import LabelEncoder
le_Grade = LabelEncoder()
df['SEX'] = le_Grade.fit_transform(df['SEX'])
df['UMUR'] = le_Grade.fit_transform(df['UMUR'])
df['TINGKAT'] = le_Grade.fit_transform(df['TINGKAT'])
df['JURUSAN'] = le_Grade.fit_transform(df['JURUSAN'])
df['MJOB'] = le_Grade.fit_transform(df['MJOB'])
df['FJOB'] = le_Grade.fit_transform(df['FJOB'])
df['Tingkat edukasi Ibu'] = le_Grade.fit_transform(df['Tingkat
edukasi Ibu'])
df['Tingkat edukasi Ayah'] = le_Grade.fit_transform(df['Tingkat
edukasi Ayah'])
df['EKSTRAKULIKULER'] =
le_Grade.fit_transform(df['EKSTRAKULIKULER'])
df['JURUSAN'] = le_Grade.fit_transform(df['JURUSAN'])
df['HIGHER'] = le_Grade.fit_transform(df['HIGHER'])
df['HEALTH'] = le_Grade.fit_transform(df['HEALTH'])
df['TRAVELTIME'] = le_Grade.fit_transform(df['TRAVELTIME'])
df['FAMTREE'] = le_Grade.fit_transform(df['FAMTREE'])
df['TINGGALDENGANORTU'] =
le_Grade.fit_transform(df['TINGGALDENGANORTU'])
df['REASON'] = le_Grade.fit_transform(df['REASON'])
df['BIMBEL'] = le_Grade.fit_transform(df['BIMBEL'])
df['INTERNET'] = le_Grade.fit_transform(df['INTERNET'])
df['ROMANTIC'] = le_Grade.fit_transform(df['ROMANTIC'])
df['HANGOUT'] = le_Grade.fit_transform(df['HANGOUT'])
df['ONLINEGAME'] = le_Grade.fit_transform(df['ONLINEGAME'])
df['SOSMED'] = le_Grade.fit_transform(df['SOSMED'])
df['STUDYTIME'] = le_Grade.fit_transform(df['STUDYTIME'])
df['FREETIME'] = le_Grade.fit_transform(df['FREETIME'])
df['ORGANISASI'] = le_Grade.fit_transform(df['ORGANISASI'])
df.head()

conditions = [
    (df['PKNS3'] >= 80),
    (df['PKNS3']<= 79)
        ]
values = ['P', 'F']
df['Score'] = np.select(conditions, values)
df.head(5)

# Tampilkan DataFrame sebelum menghapus fitur
print("DataFrame Sebelum:")
print(df)
```

11. *Support Vector Machine* PKn

```
fitur_yang_dihapus = ['MATEMATIKAS3','INGGRISS3', 'Nama',
'TINGKAT', 'JURUSAN','UMUR', 'SEX', 'PKNS3', 'BINDONESIAS3']
df = df.drop(columns=fitur_yang_dihapus)

# Tampilkan DataFrame setelah menghapus fitur
print("\nDataFrame Setelah:")
print(df)

#jumlah data 1 dan 0 pada target
df['Score'].value_counts()

inputs = df.drop('Score',axis='columns')
target = df['Score']
inputs.head()

#Pembagian data
from sklearn.model_selection import train_test_split # Import
train_test_split function
pred_train, pred_test, tar_train, tar_test =
train_test_split(inputs, target, test_size=0.4, random_state=0)

# finding best parameter for SVM
param_grid = {
  'C': [0.1, 1, 10, 100, 1000],
  'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
   'kernel': ['poly', 'rbf', 'sigmoid']
}

from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
grid = GridSearchCV(SVC(), param_grid, refit=True, verbose=3)
grid.fit(pred_train,tar_train)
grid.best_params_

model = svm.SVC(C= 100, gamma= 0.0001)
classifier = model.fit(pred_train,tar_train)
classifier

predictions = classifier.predict(pred_test)
predictions

from sklearn.metrics import classification_report
print(classification_report(tar_test, predictions))

#Get the confusion matrix
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt

cf_matrix = confusion_matrix(tar_test, predictions)
print(cf_matrix)

import seaborn as sns
sns.heatmap(cf_matrix, annot=True)

ax = sns.heatmap(cf_matrix, annot=True,fmt="d", cmap='Blues')

ax.set_title('PKN SVM Confusion Matrix 60:40');
ax.set_xlabel('Predictions')
```

```
ax.set_ylabel('Actuals ');

ax.xaxis.set_ticklabels(['Failed', 'Passed'])
ax.yaxis.set_ticklabels(['Failed', 'Passed'])
plt.show()

from sklearn.metrics import matthews_corrcoef
mcc_score = matthews_corrcoef(tar_test, predictions)
print("Matthews Correlation Coefficient:", mcc_score)
```

12. *Naïve Bayes* PKn

```
import numpy as np
import pandas as pd
# To calculate the accuracy score of the model
from sklearn.metrics import accuracy_score
from sklearn.naive_bayes import GaussianNB

df = pd.read_excel(r'''C:\Users\RAPHAEL
KUSUMO\OneDrive\Documents\Skripsi 2023 Juli\SORTER DATA
GFORM4.XLSX''')
df.head()

from sklearn.preprocessing import LabelEncoder
le_Grade = LabelEncoder()
df['SEX'] = le_Grade.fit_transform(df['SEX'])
df['UMUR'] = le_Grade.fit_transform(df['UMUR'])
df['TINGKAT'] = le_Grade.fit_transform(df['TINGKAT'])
df['JURUSAN'] = le_Grade.fit_transform(df['JURUSAN'])
df['MJOB'] = le_Grade.fit_transform(df['MJOB'])
df['FJOB'] = le_Grade.fit_transform(df['FJOB'])
df['Tingkat edukasi Ibu'] = le_Grade.fit_transform(df['Tingkat
edukasi Ibu'])
df['Tingkat edukasi Ayah'] = le_Grade.fit_transform(df['Tingkat
edukasi Ayah'])
df['EKSTRAKULIKULER'] =
le_Grade.fit_transform(df['EKSTRAKULIKULER'])
df['JURUSAN'] = le_Grade.fit_transform(df['JURUSAN'])
df['HIGHER'] = le_Grade.fit_transform(df['HIGHER'])
df['HEALTH'] = le_Grade.fit_transform(df['HEALTH'])
df['TRAVELTIME'] = le_Grade.fit_transform(df['TRAVELTIME'])
df['FAMTREE'] = le_Grade.fit_transform(df['FAMTREE'])
df['TINGGALDENGANORTU'] =
le_Grade.fit_transform(df['TINGGALDENGANORTU'])
df['REASON'] = le_Grade.fit_transform(df['REASON'])
df['BIMBEL'] = le_Grade.fit_transform(df['BIMBEL'])
df['INTERNET'] = le_Grade.fit_transform(df['INTERNET'])
df['ROMANTIC'] = le_Grade.fit_transform(df['ROMANTIC'])
df['HANGOUT'] = le_Grade.fit_transform(df['HANGOUT'])
df['ONLINEGAME'] = le_Grade.fit_transform(df['ONLINEGAME'])
df['SOSMED'] = le_Grade.fit_transform(df['SOSMED'])
df['STUDYTIME'] = le_Grade.fit_transform(df['STUDYTIME'])
df['FREETIME'] = le_Grade.fit_transform(df['FREETIME'])
df['ORGANISASI'] = le_Grade.fit_transform(df['ORGANISASI'])
df.head()
```

```
conditions = [
    (df['PKNS3'] >= 80),
    (df['PKNS3']<= 79)
        ]
values = ['P', 'F']
df['Score'] = np.select(conditions, values)
df.head(5)

# Tampilkan DataFrame sebelum menghapus fitur
print("DataFrame Sebelum:")
print(df)

# Menghapus beberapa fitur (kolom)
fitur_yang_dihapus = ['MATEMATIKAS3','INGGRISS3', 'Nama',
'TINGKAT', 'JURUSAN','UMUR', 'SEX', 'PKNS3', 'BINDONESIAS3']
df = df.drop(columns=fitur_yang_dihapus)

# Tampilkan DataFrame setelah menghapus fitur
print("\nDataFrame Setelah:")
print(df)

#jumlah data 1 dan 0 pada target
df['Score'].value_counts()

inputs = df.drop('Score',axis='columns')
target = df['Score']
inputs.head()

#Pembagian data
from sklearn.model_selection import train_test_split # Import
train_test_split function
pred_train, pred_test, tar_train, tar_test =
train_test_split(inputs, target, test_size=0.4, random_state=0)

from sklearn.ensemble import BaggingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Create a Gaussian Naive Bayes classifier
gnb_classifier = GaussianNB()

# Create a BaggingClassifier with Gaussian Naive Bayes as the
base estimator
bagging_classifier =
BaggingClassifier(base_estimator=gnb_classifier,
n_estimators=10, random_state=0)

# Train the BaggingClassifier on the training data
bagging_classifier.fit(pred_train, tar_train)

# Make predictions on the test data
y_pred = bagging_classifier.predict(pred_test)

# Calculate accuracy
accuracy = accuracy_score(tar_test, y_pred)
print("Accuracy:", accuracy)
```

```
from sklearn.metrics import classification_report
print(classification_report(tar_test, y_pred))

#Get the confusion matrix
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt

cf_matrix = confusion_matrix(tar_test, y_pred)
print(cf_matrix)

import seaborn as sns
sns.heatmap(cf_matrix, annot=True)

ax = sns.heatmap(cf_matrix, annot=True,fmt="d", cmap='Blues')

ax.set_title('PKN NB Confusion Matrix 60:40');
ax.set_xlabel('Predictions')
ax.set_ylabel('Actuals ');

ax.xaxis.set_ticklabels(['Failed', 'Passed'])
ax.yaxis.set_ticklabels(['Failed', 'Passed'])
plt.show()

from sklearn.metrics import matthews_corrcoef
mcc_score = matthews_corrcoef(tar_test, y_pred)
print("Matthews Correlation Coefficient:", mcc_score)
```

LAMPIRAN D

Variabel Penelitian

Tabel D.1 Variabel Penelitian

No	Variabel	Penjelasan
1	*Sex*	Jenis kelamin siswa (laki-laki atau perempuan)
2	*Age*	Umur siswa (numerik)
3	Medu	Tingkat edukasi ibu 1= tidak ada 2= lulus SD 3= lulus SMP 4= lulus SMA 5= lulusan D3 dan pendidikan yang lebih tinggi.
4	Fedu	Tingkat edukasi ibu 1= tidak ada 2= lulus SD 3= lulus SMP 4= lulus SMA 5= lulusan D3 dan pendidikan yang lebih tinggi.
5	*Studytime*	Waktu belajar siswa dalam satu minggu (numerik). Nilai disini dinyatakan dalam waktu jam.
6	Famrel	Hubungan dalam keluarga (1-5). Nilai 1 menandakan sangat tidak dekat dan nilai 5 menandakan sangat dekat dengan keluarga.
7	*Freetime*	Waktu luang (numerik). Nilai waktu luang disini dinyatakan dalam jam.
8	*Health*	Tingkat kesehatan siswa (1-5). Nilai 1 menandakan siswa sangat sakit dan nilai 5 menandakan siswa sehat.
9	Famsize	Jumlah anggota keluarga (lebih kecil dari 3 atau lebih dari tiga)
10	Pstatus	Status tinggal dengan orang tua (ya atau tidak)
11	Mjob	Pekerjaan Ibu. 1= ibu rumah tangga 2= guru 3= pegawai swasta 4= pns 5= Pekerjaan lainya (pilih pilihan ini jika tidak termasuk pekerjaan yang disebutkan di atas.
12	Fjob	Pekerjaan Ayah Pekerjaan Ibu. 1= ibu rumah tangga 2= guru 3= pegawai swasta 4= PNS 5= Pekerjaan lainya (pilih pilihan ini jika tidak termasuk pekerjaan yang disebutkan

No	Variabel	Penjelasan
		di atas.
13	*reason*	Alasan memilih sekolah ini 1= dekat dengan rumah 2= reputasi sekolah 3= pembelajaran yang baik 4= alasan lainya.
14	schoolsup	Mengikuti pembelajaran tambahan seperti bimbel di luar sekolah (ya atau tidak)
15	*Traveltime*	Waktu perjalanan dari rumah ke sekolah (numerik).
16	Famsud	Biaya dukungan dari keluarga (ya atau tidak
17	*Activities*	Mengikuti ekstrakurikuler (ya atau tidak)
18	*Higher*	Apakah ingin melanjutkan pendidikan ke jenjang yang lebih tinggi (ya atau tidak)
19	*Internet*	Akses internet di rumah (ya atau tidak)
20	*Romantic*	Memiliki hubungan dengan lawan jenis (ya atau tidak)
21	Goout	Apakah sering bermain dengan teman (ya atau tidak)
22	Sosmed	Jumlah waktu yang digunakan untuk membuka sosial media. 0 = 1 jam 1 = 2 jam 2 = 3 jam 3 = lebih dari 3 jam
23	Onlinegame	Jumlah waktu yang digunakan untuk bermain game online. 0 = 1 jam 1 = 2 jam 2 = 3 jam 3 = lebih dari 3 jam
22	Fisikas2	Nilai Fisika semester 2
23	Kimias2	Nilai Kimia semester 2
24	PKNs2	Nilai PKn semester 2
25	Binggriss2	Nilai Bahasa Inggris semester 2
26	Matematikas2	Nilai matematika semester 2
27	PKNs3 (Target data)	Nilai PKn semester 3
28	Binggriss3 (Target data)	Nilai Bahasa Inggris semester 3
29	Matematikas3 (Target data)	Nilai Matematika semester 3

Variabel yang digunakan pada penelitian ini merupakan variabel yang bisa digunakan untuk membuat prediksi performa siswa. variabel dari faktor ekonomi. faktor keluarga, jenis kelamin, dan umur bisa digunakan untuk membuat prediksi performa siswa [9] [13]. Variabel nilai siswa pada mata pelajaran semester sebelumnya juga bisa digunakan untuk membuat prediksi performa siswa [9] [11] [12]. Seluruh variabel yang ada pada Tabel D.1 diatas merupakan variabel yang

dinyatakan memiliki korelasi dengan performa siswa pada mata pelajaran Bahasa Inggris dan Matematika yang dibuktikan dengan penelitian-penelitian sebelumnya sudah menggunakan variabel-variabel di atas [8-14]. Variabel-variabel di atas juga sudah dilakukan ujicoba dalam pembuatan model prediksi performa siswa dan hasilnya variabel-variabel tersebut memang memiliki pengaruh pada prediksi performa siswa.

LAMPIRAN E

Perancangan Algoritma

Perancangan arsitektur model *machine learning* pada penelitian ini secara singkat dapat dilihat pada Bab 3.1 tentang perancangan penelitian dan pada Gambar 3.1. Perancangan model secara lebih detil dapat dilihat pada penjelasan di bawah ini.

A. Perancangan Model DT

Perencangan model *machine learning* pada algoritma DT untuk mata pelajaran Matematika dan Bahasa Inggris menggunakan tahapan perancangan sebagai berikut.

1. Melakukan pembacaan data penelitian.
2. Melakukan Transformasi data dari tipe *string* ke *integer*.
3. Melakukan pembagian *training* dan testing data.
4. Membuat sistem *tuning* parameter untuk mencari parameter terbaik dari model DT.
5. Pada algoritma DT, hasil dari *tuning* parameter yang dilakukan menghasilkan pengaturan parameter di bawah ini.

   A. max_depth dengan nilai max_depth=150. Ini artinya bahwa kedalaman dari *tree* akan dibuat sampai kedalaman maksimal 150.

   B. Max_features=log2. Parameter max_features ini mengendalikan berapa banyak variabel yang digunakan ketika mencari split terbaik dari *tree* yang dibuat. Nilai log2 ini menandakan bahwa dalam pencarian *split* terbaik menggunakan rumus log2(x_variabel) yang dimana pada program penelitian ini x_variabel yang ada berjumlah 26 sehingga rumusnya menjadi log2(26).

   C. min_samples_split=16. Parameter ini menentukan jumlah *sample* data yang diperlukan untuk melakukan *split* dari *internal node*. *Internal node* sendiri adalah *node* yang membagi data berdasarkan variabel atau kondisi tertentu.

   D. random_state=0. Parameter ini mengontrol pengacakan *training* data dalam membuat model DT. Parameter ini perlu diatur agar program yang dijalankan menghasilkan nilai yang sama setiap kali dijalankan.

   E. Criterion=gini. Gini *impurity* mengukur probabilitas ketika sebuah sampel acak yang dipilih secara salah satu kelas secara acak. Nilai Gini

impurity semakin rendah jika sampel yang dipilih secara acak cenderung memiliki label yang sama, dan semakin tinggi jika labelnya tersebar merata di antara kelas-kelas yang ada.

6. Setelah didapatkan hasil parameter terbaik, selanjutnya parameter terbaik tersebut digunakan untuk membuat model *machine learning* yang dilatih dengan *training* data untuk setiap algoritma yang digunakan. Pada model ini digunakan juga baggingclassifier yang membuat model di atas menjadi 10 model *tree*. Setiap *tree* ini akan membuat prediksi masing-masing dan hasilnya prediksi akhirnya akan ditentukan berdasarkan voting dari 10 *tree* yang sudah dibuat sebelumnya.
7. Hasil dari setiap model di atas kemudian digunakan untuk membuat prediksi dari testing data. Hasil dari model ini juga menghasilkan *confusion matrix*, *accuracy*, *precision*, *recall*, f1score, dan MCC.

Perencangan model *machine learning* pada algoritma DT untuk mata pelajaran PKn menggunakan parameter yang berbeda dengan model DT pada mata pelajaran Matematika dan Bahasa Inggris. Tahapan perancangan DT pada mata pelajaran PKn adalah sebagai berikut.

1. Melakukan pembacaan data penelitian.
2. Melakukan Transformasi data dari tipe *string* ke *integer*.
3. Melakukan pembagian *training* dan testing data.
4. Membuat sistem *tuning* parameter untuk mencari parameter terbaik dari model DT.
5. Pada algoritma DT, hasil dari *tuning* parameter yang dilakukan menghasilkan pengaturan parameter di bawah ini.

   A. max_depth dengan nilai max_depth=100. Ini artinya bahwa kedalaman dari *tree* akan dibuat sampai kedalaman maksimal 100.

   B. Max_features=sqrt. Parameter max_features ini mengendalikan berapa banyak variabel yang digunakan ketika mencari split terbaik dari *tree* yang dibuat. Nilai sqrt ini menandakan bahwa dalam pencarian *split* terbaik menggunakan rumus sqrt (sqrt adalah akar pangkat 2) dari x_variabel yang

dimana pada program penelitian ini x_variabel yang ada berjumlah 26 sehingga rumusnya menjadi sqrt(26).

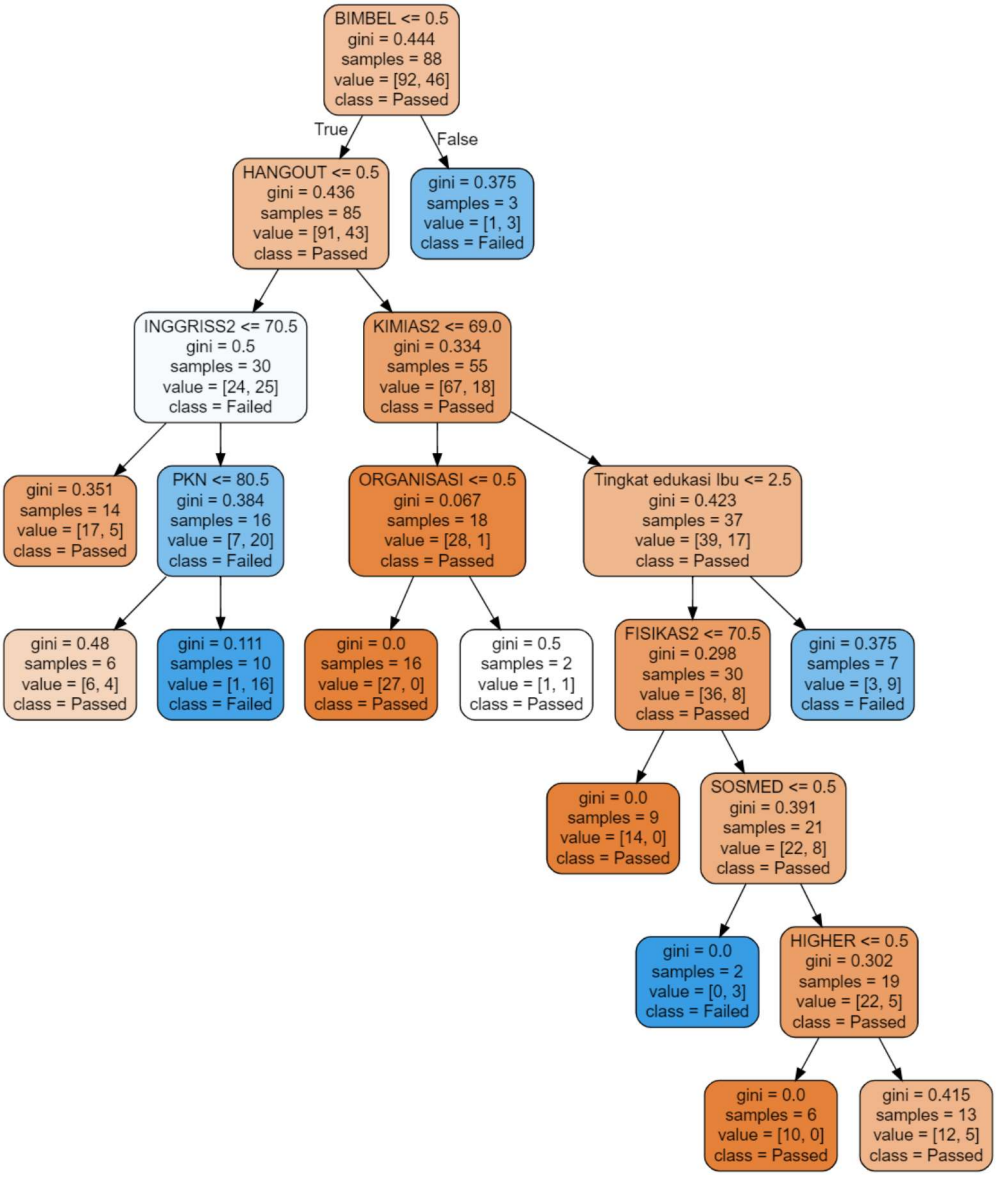
C. min_samples_split=18. Parameter ini menentukan jumlah *sample* data yang diperlukan untuk melakukan *split* dari *internal node*. *Internal node* sendiri adalah *node* yang membagi data berdasarkan variabel atau kondisi tertentu.

D. random_state=0. Parameter ini mengontrol pengacakan *training* data dalam membuat model DT. Parameter ini perlu diatur agar program yang dijalankan menghasilkan nilai yang sama setiap kali dijalankan.

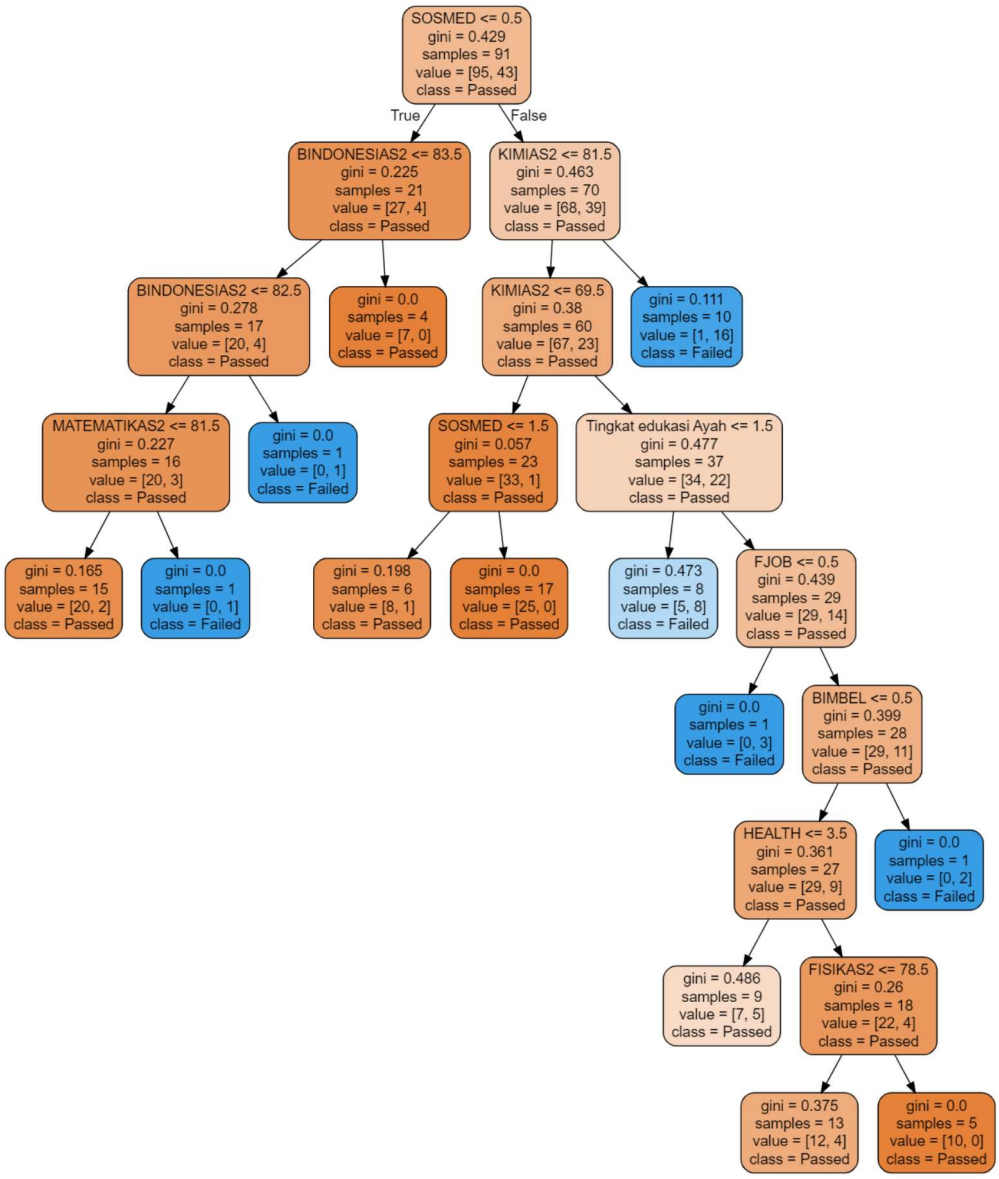
E. Criterion=gini. Gini *impurity* mengukur probabilitas ketika sebuah sampel acak yang dipilih secara salah satu kelas secara acak. Nilai Gini impurity semakin rendah jika sampel yang dipilih secara acak cenderung memiliki label yang sama, dan semakin tinggi jika labelnya tersebar merata di antara kelas-kelas yang ada.

6. Setelah didapatkan hasil parameter terbaik, selanjutnya parameter terbaik tersebut digunakan untuk membuat model *machine learning* yang dilatih dengan *training* data untuk setiap algoritma yang digunakan. Pada model ini digunakan juga baggingclassifier yang membuat model di atas menjadi 10 model *tree*. Setiap *tree* ini akan membuat prediksi masing-masing dan hasilnya prediksi akhirnya akan ditentukan berdasarkan voting dari 10 tree yang sudah dibuat sebelumnya.
7. Hasil dari setiap model di atas kemudian digunakan untuk membuat prediksi dari testing data. Hasil dari model ini juga menghasilkan *confusion matrix*, *accuracy*, *precision*, *recall*, f1score, dan MCC.

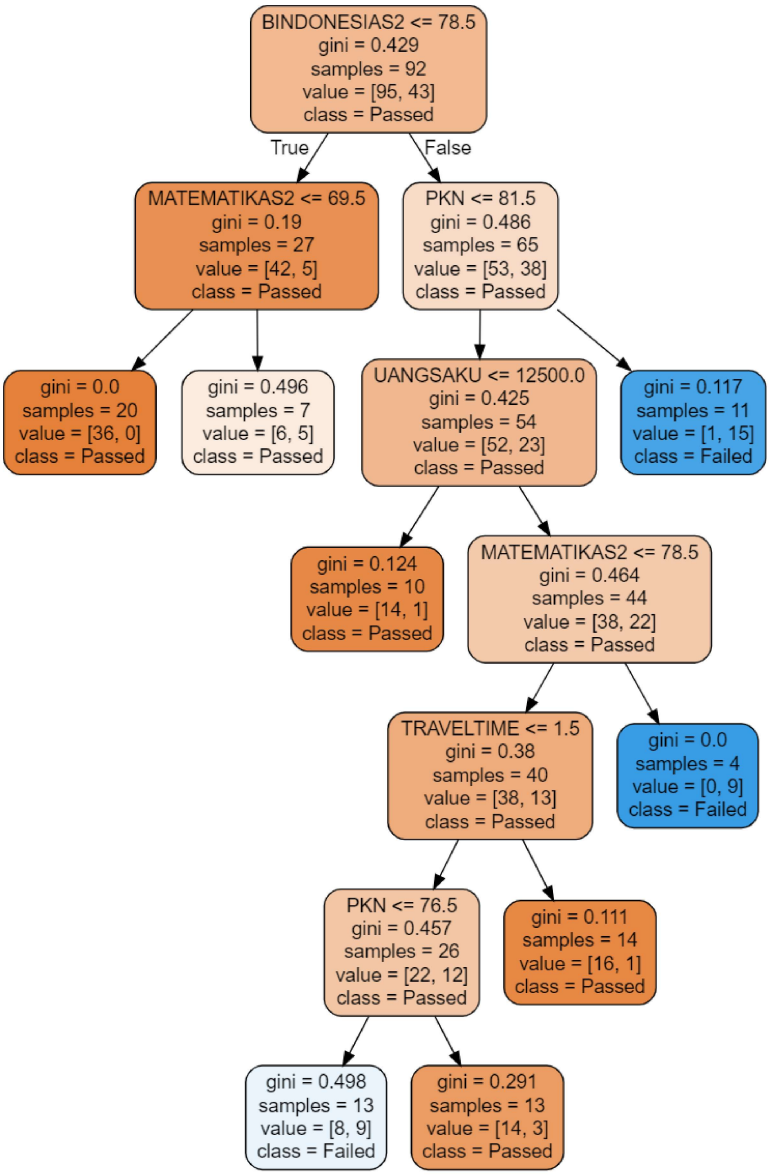
Hasil dari model DT di atas juga bisa digambarkan pohon keputusanya. DT memang merupakan algoritma yang paling mudah untuk divisualisasikan. Hasil pohon Keputusan untuk mata pelajaran Matematika dapat dilihat pada Gambar E.1 sampai Gambar E.10. Hasil pohon Keputusan untuk mata pelajaran Bahasa Inggris dapat dilihat pada Gambar E.11 sampai Gambar E.20. Hasil pohon Keputusan untuk mata pelajaran PKn dapat dilihat pada Gambar E.21 sampai Gambar E.30 yang hasilnya dapat dilihat pada gambar di bawah ini.



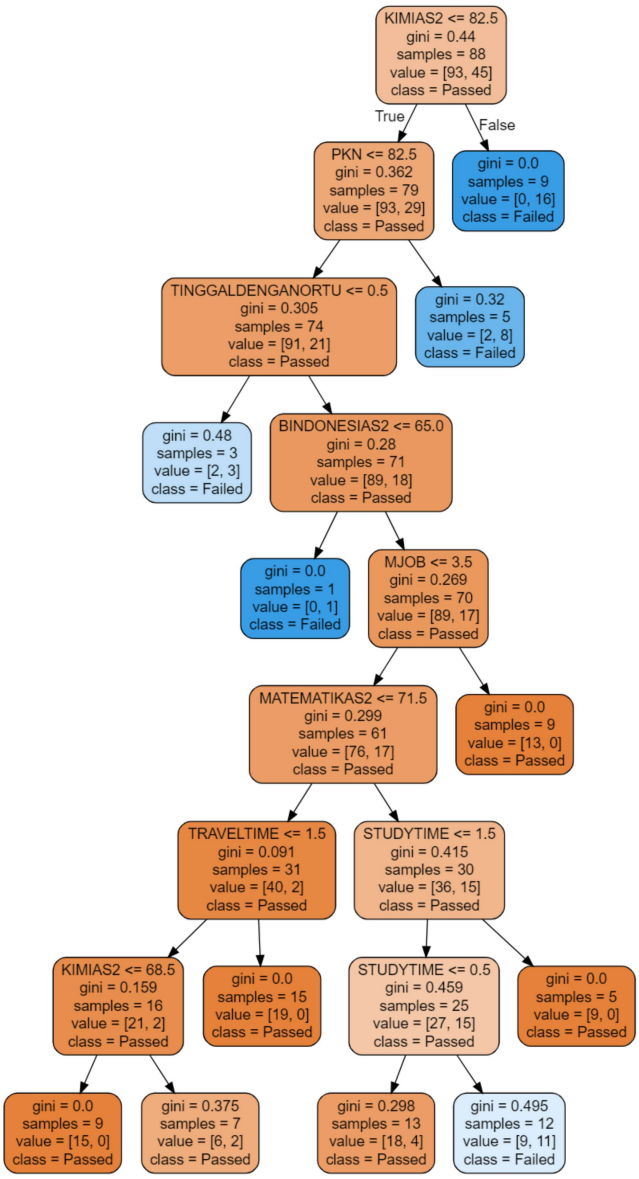
Gambar E.1 Model DT Matematika Pohon Keputusan 1



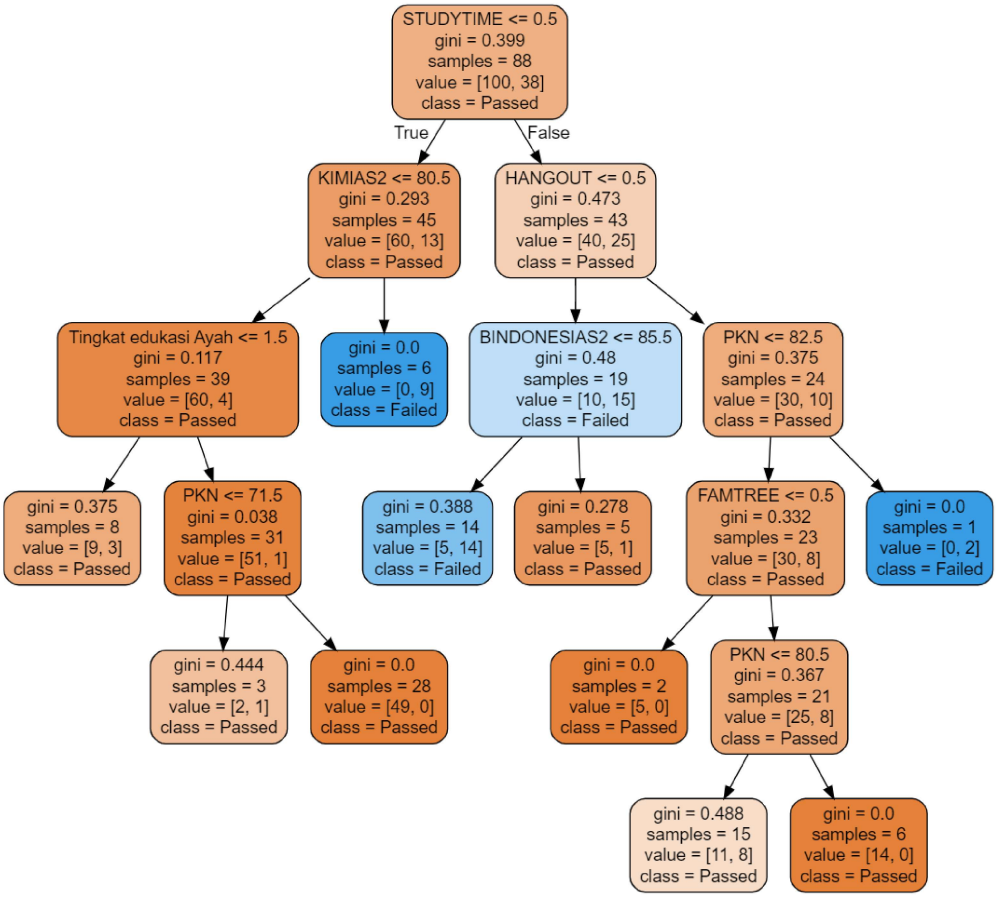
Gambar E.2 Model DT Matematika Pohon Keputusan 2



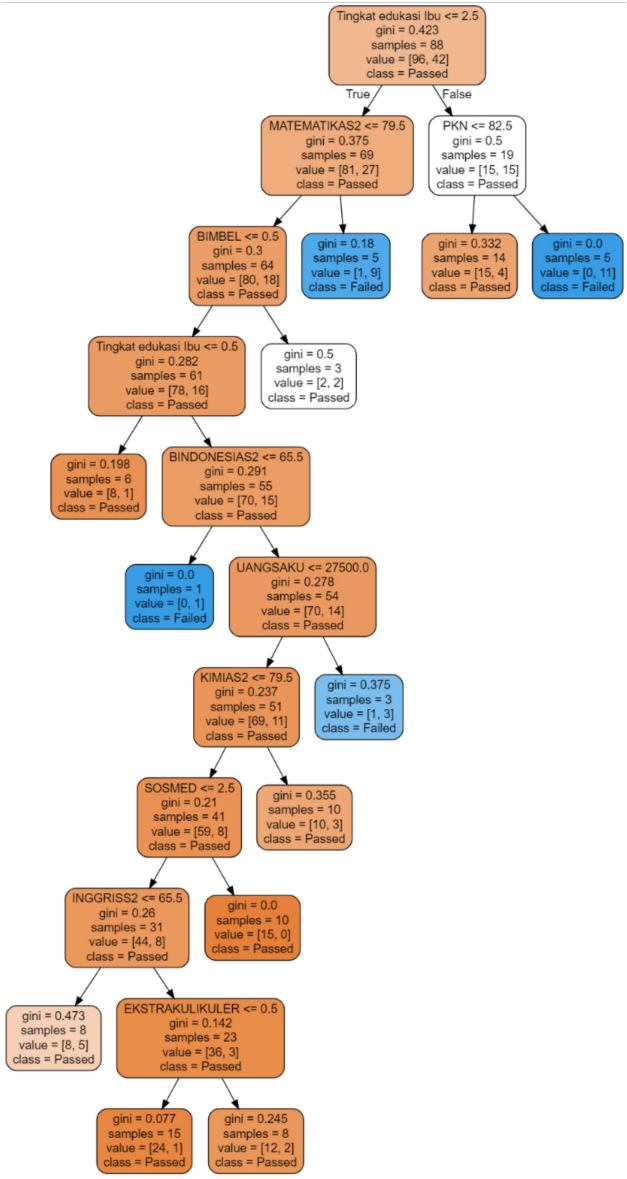
Gambar E.3 Model DT Matematika Pohon Keputusan 3



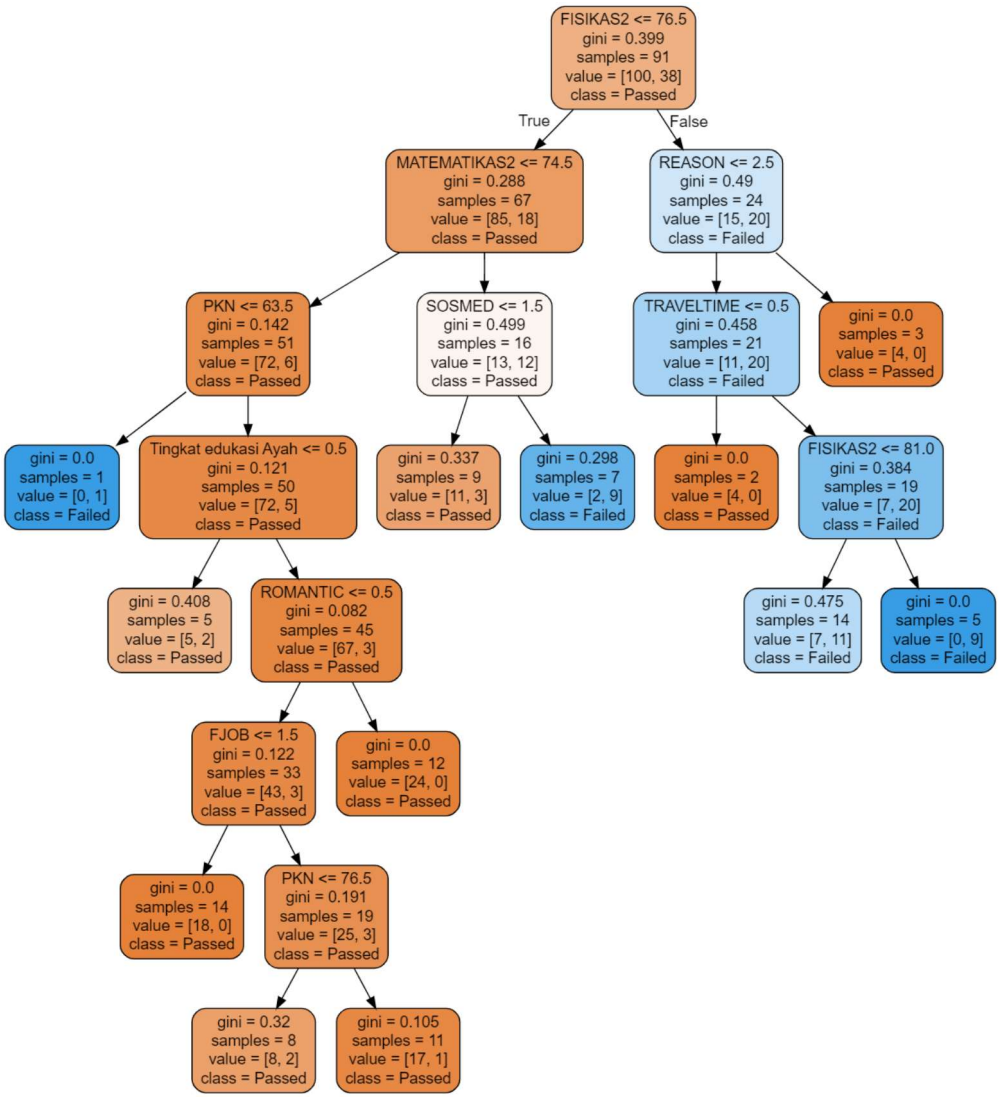
Gambar E.4 Model DT Matematika Pohon Keputusan 4



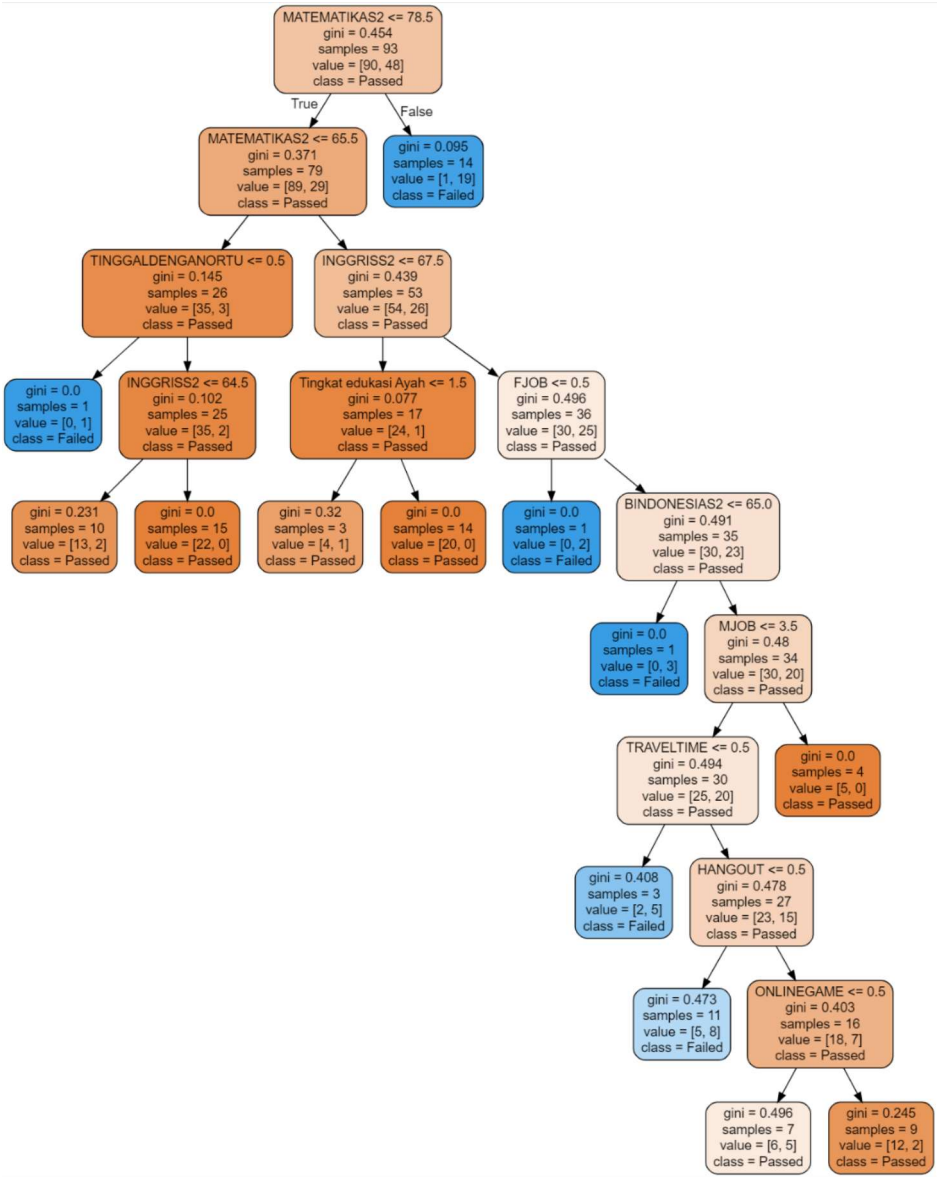
Gambar E.5 Model DT Matematika Pohon Keputusan 5



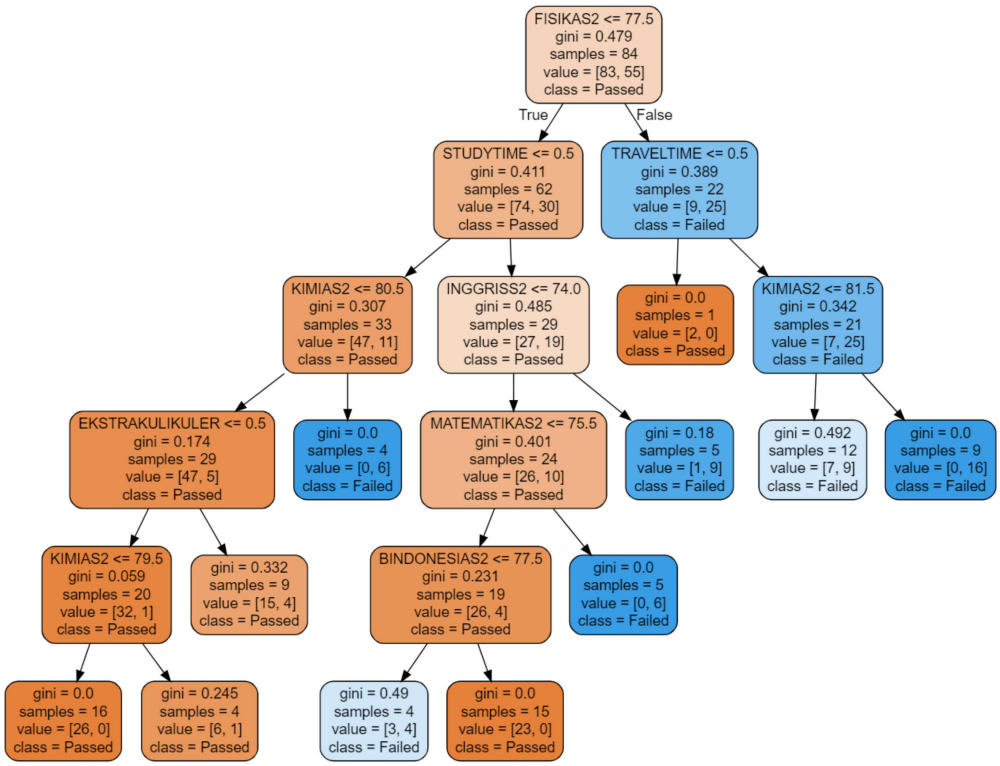
Gambar E.6 Model DT Matematika pohon Keputusan 6



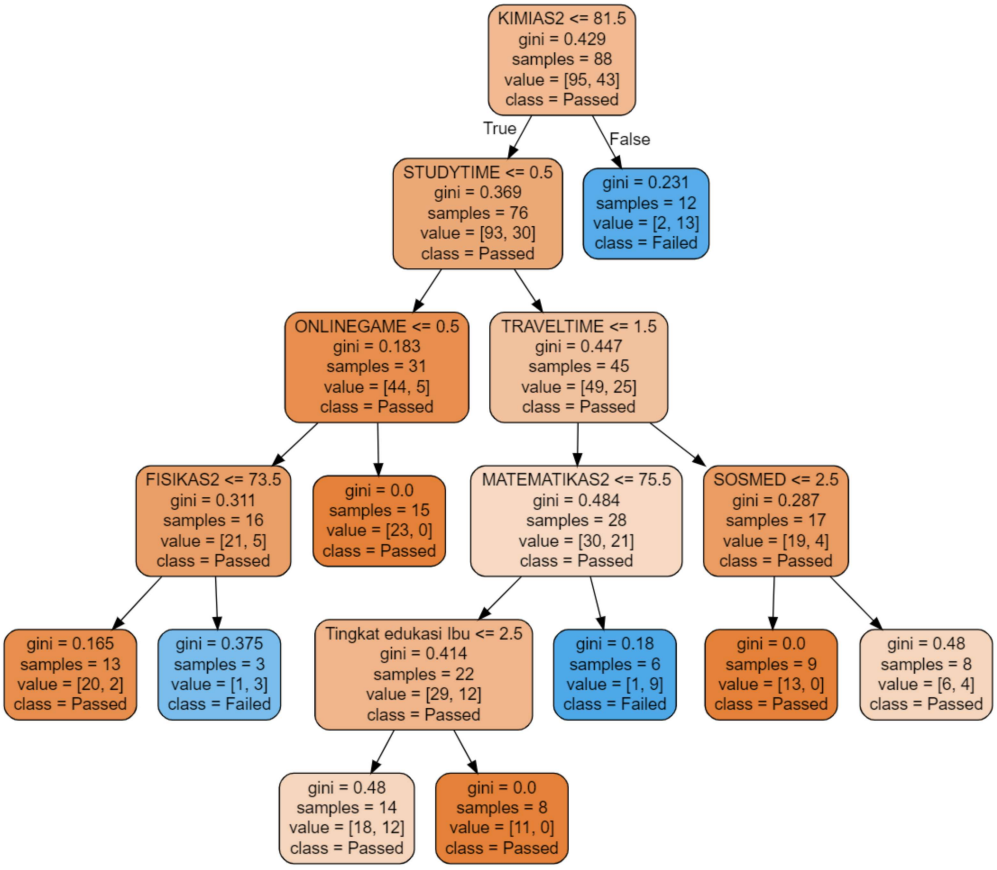
Gambar E.7 Model DT Matematika Pohon Keputusan 7



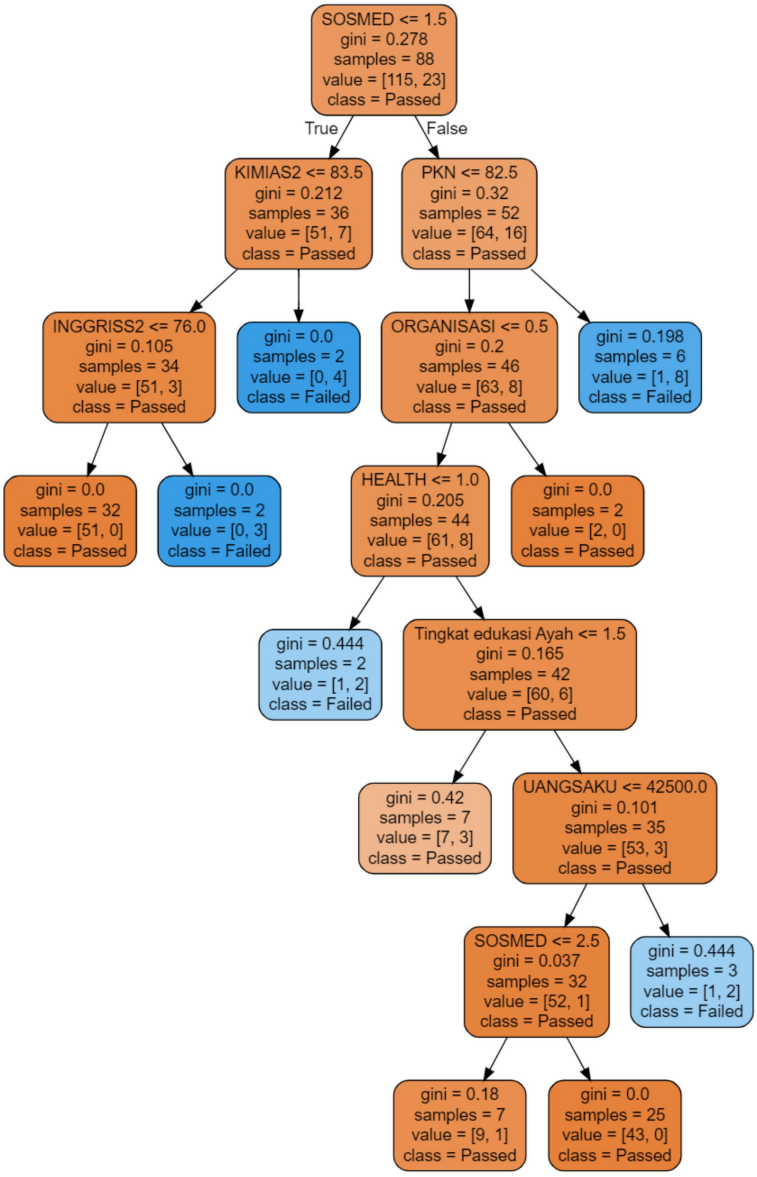
Gambar E.8 Model DT Matematika Pohon Keputusan 8



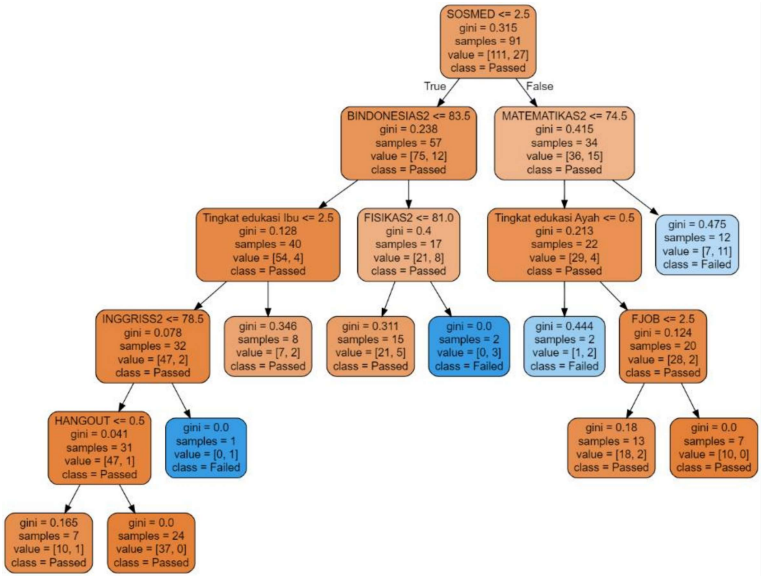
Gambar E.9 Model DT Matematika Pohon Keputusan 9



Gambar E.10 Model DT Matematika Pohon Keputusan 10

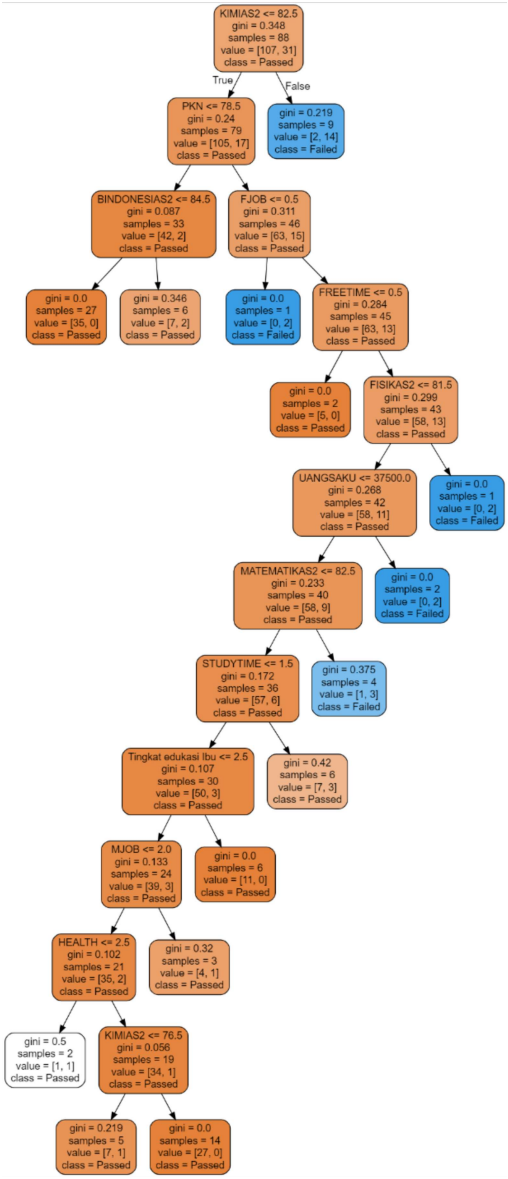


Gambar E.11 Model DT Bahasa Inggris Pohon Keputusan 1

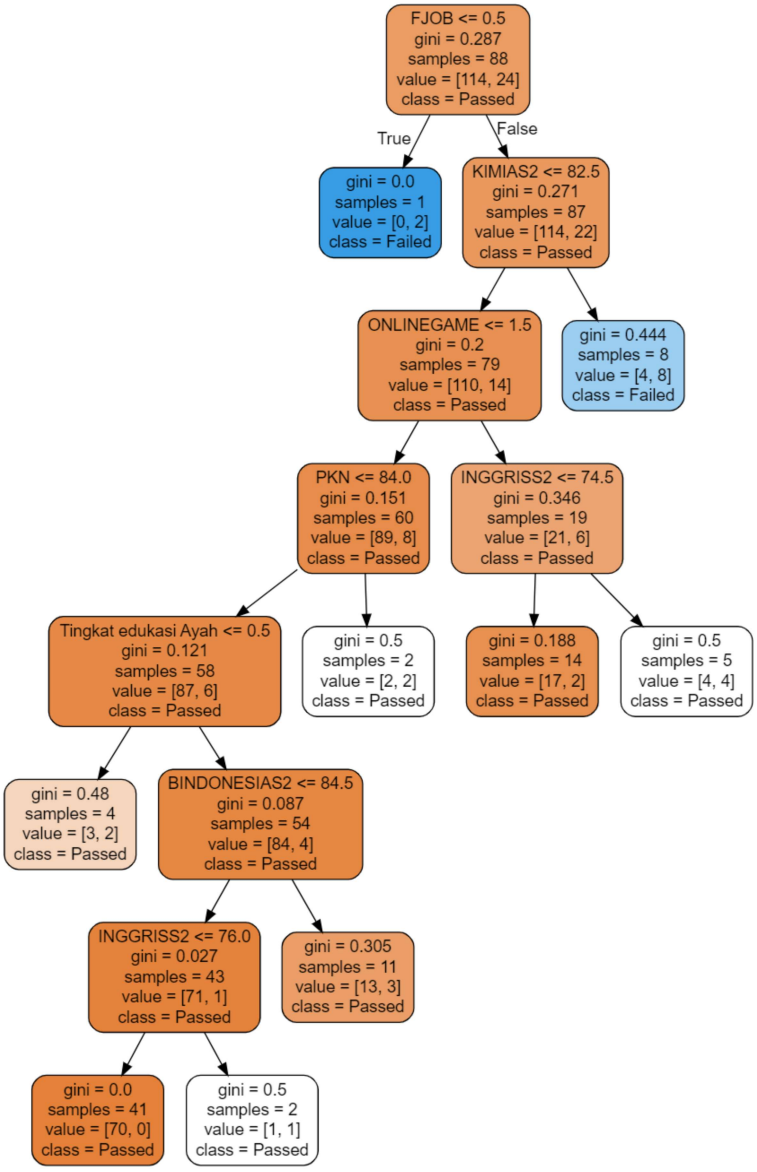


Gambar E.12 Model DT Bahasa Inggris Pohon Keputusan 2

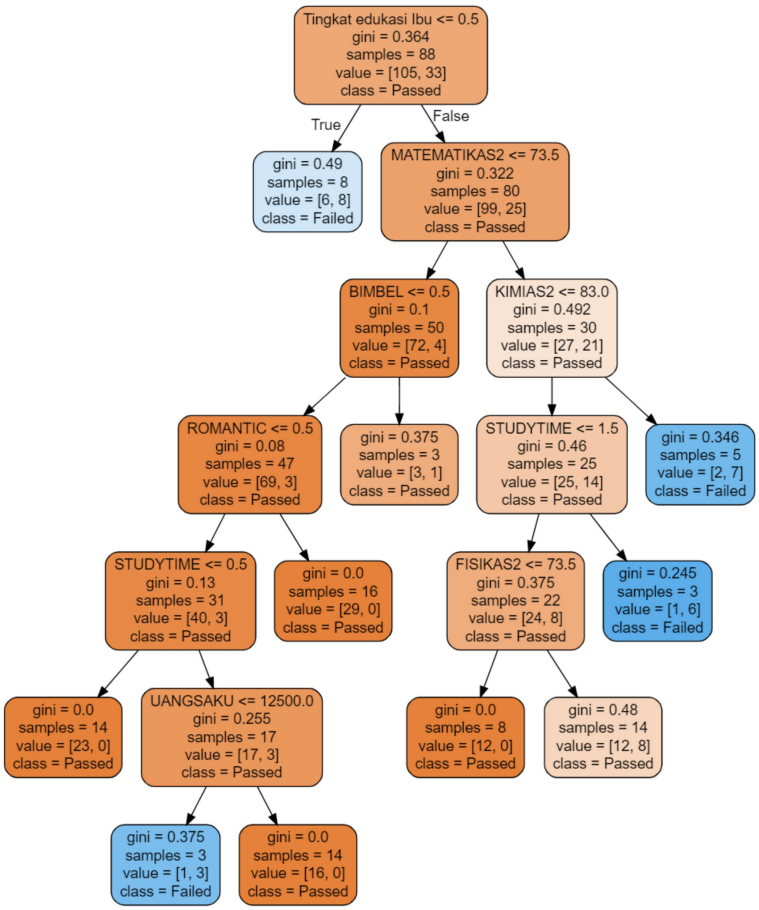
Gambar E.13 Model DT Bahasa Inggris Pohon Keputusan 3



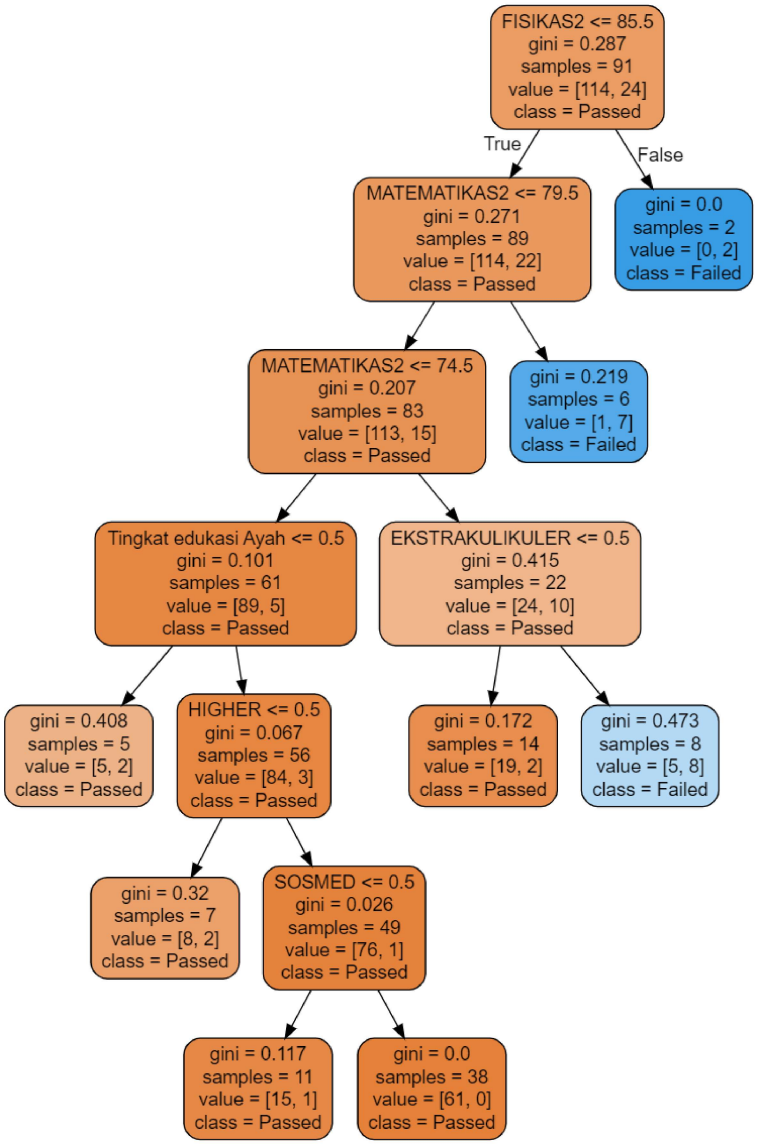
Gambar E.14 Model DT Bahasa Inggris Pohon Keputusan 4



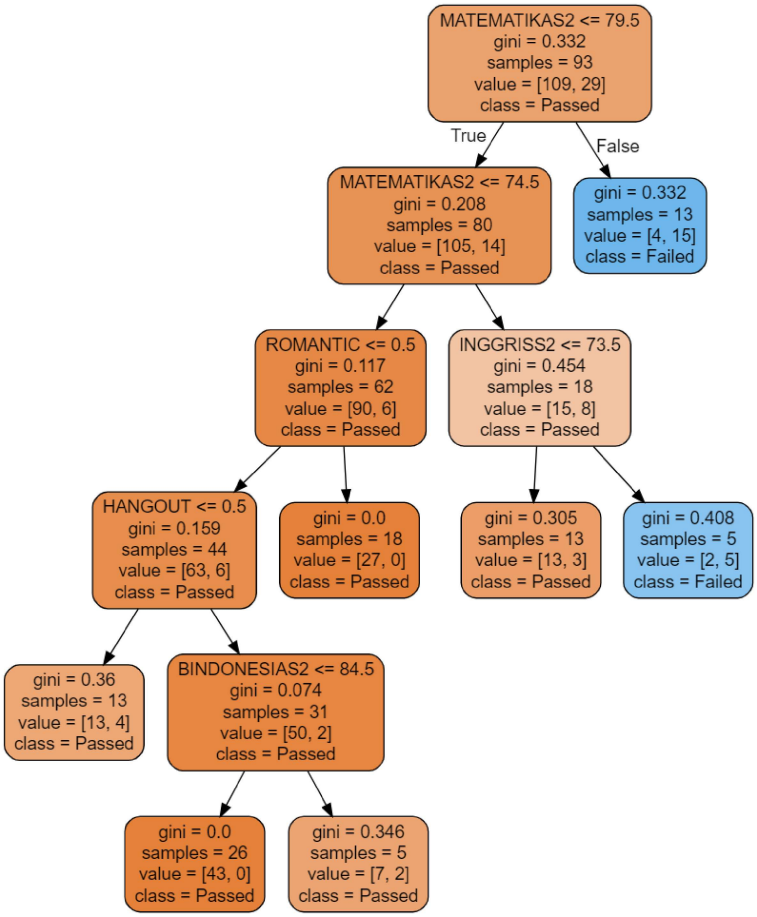
Gambar E.15 Model DT Bahasa Inggris Pohon Keputusan 5



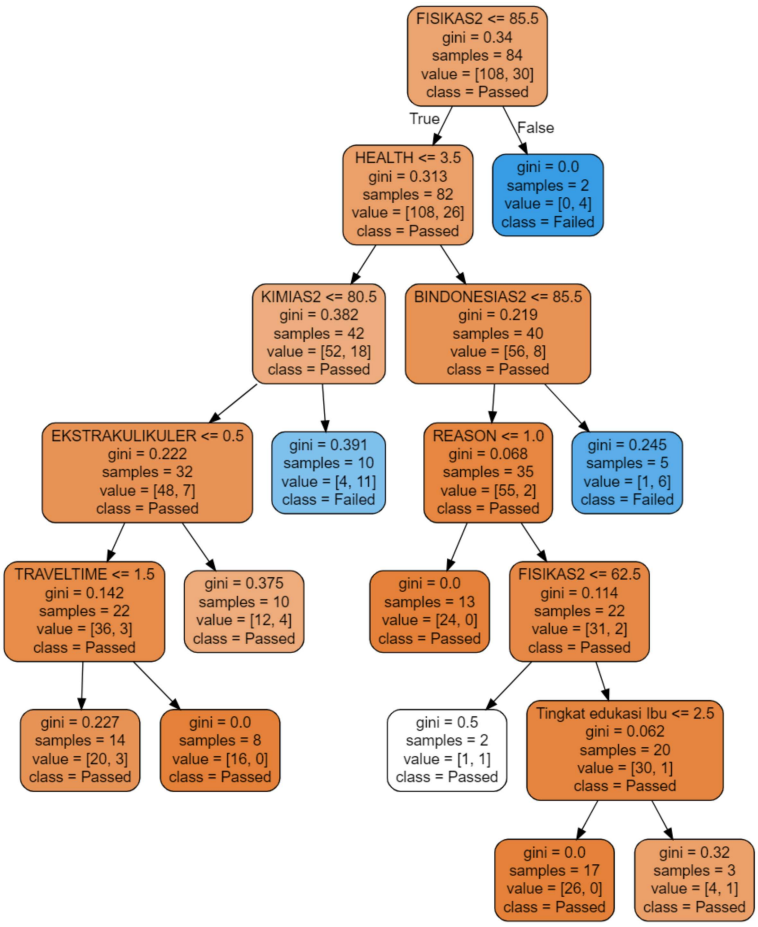
Gambar E.16 Model DT Bahasa Inggris Pohon Keputusan 6



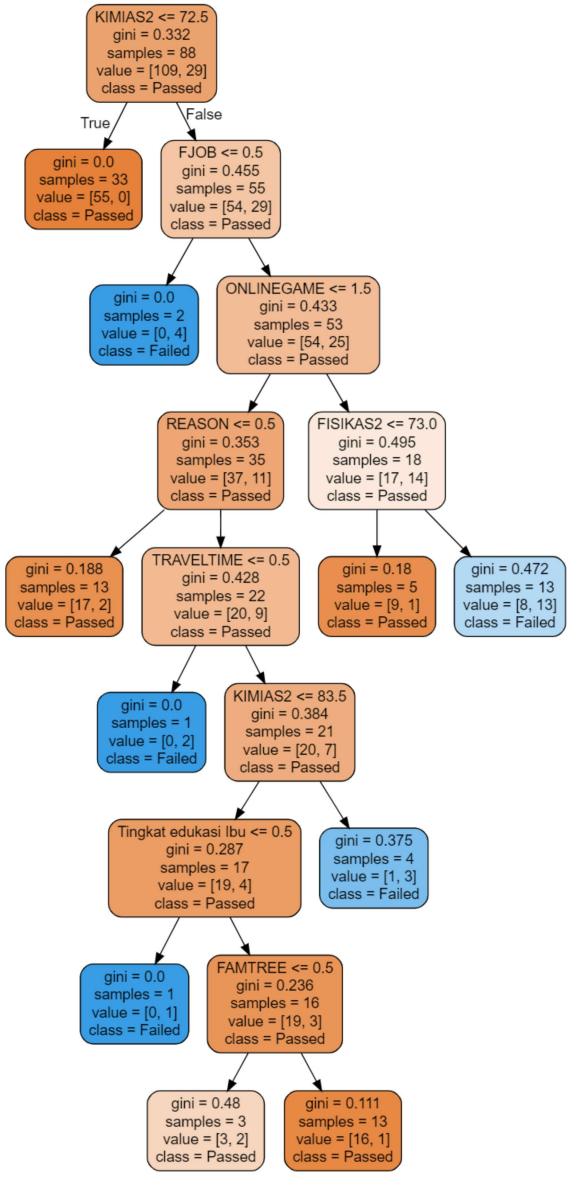
Gambar E.17 Model DT Bahasa Inggris Pohon Keputusan 7



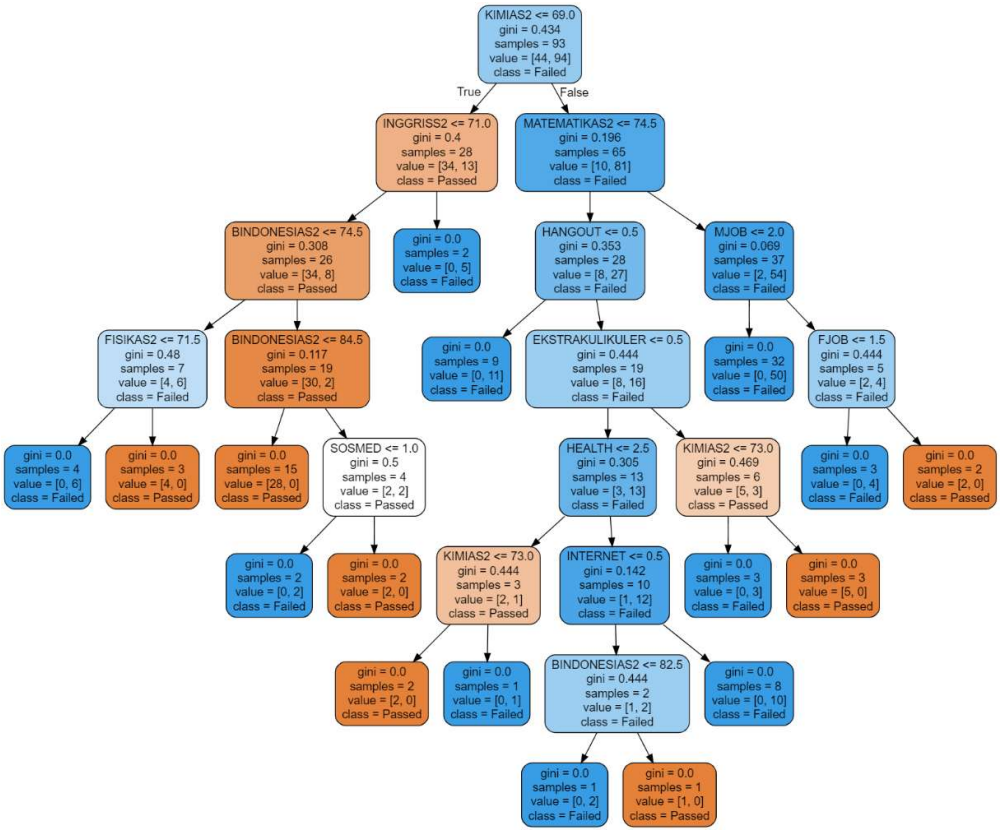
Gambar E.18 Model DT Bahasa Inggris Pohon Keputusan 8



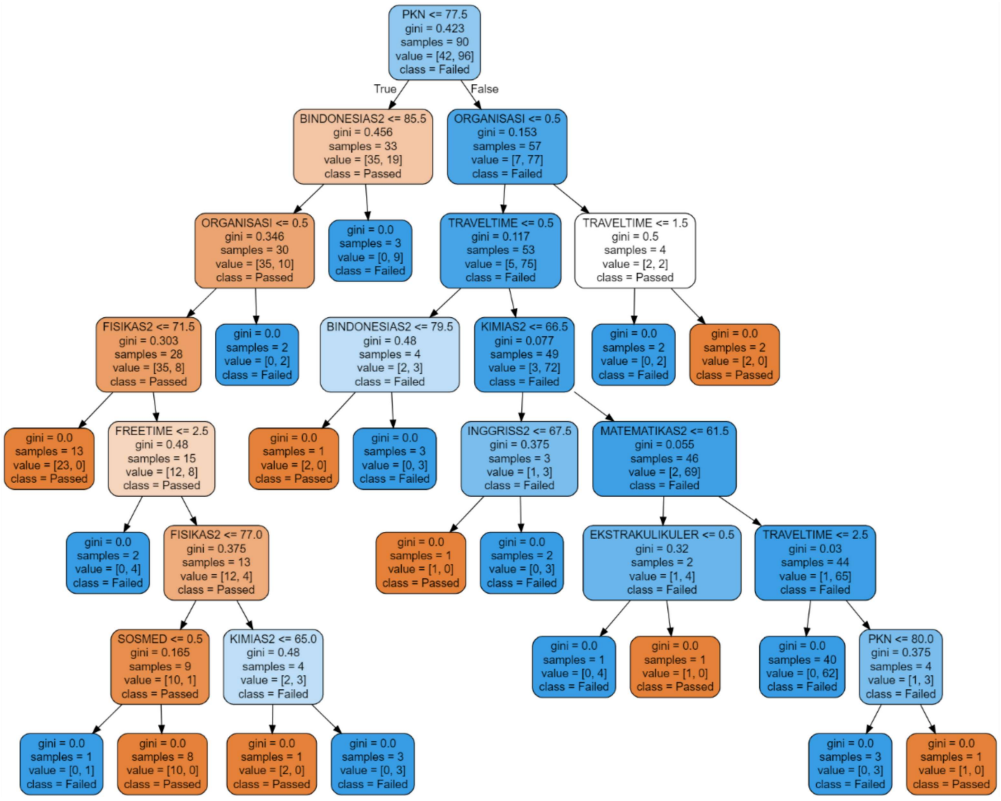
Gambar E.19 Model DT Bahasa Inggris Pohon Keputusan 9



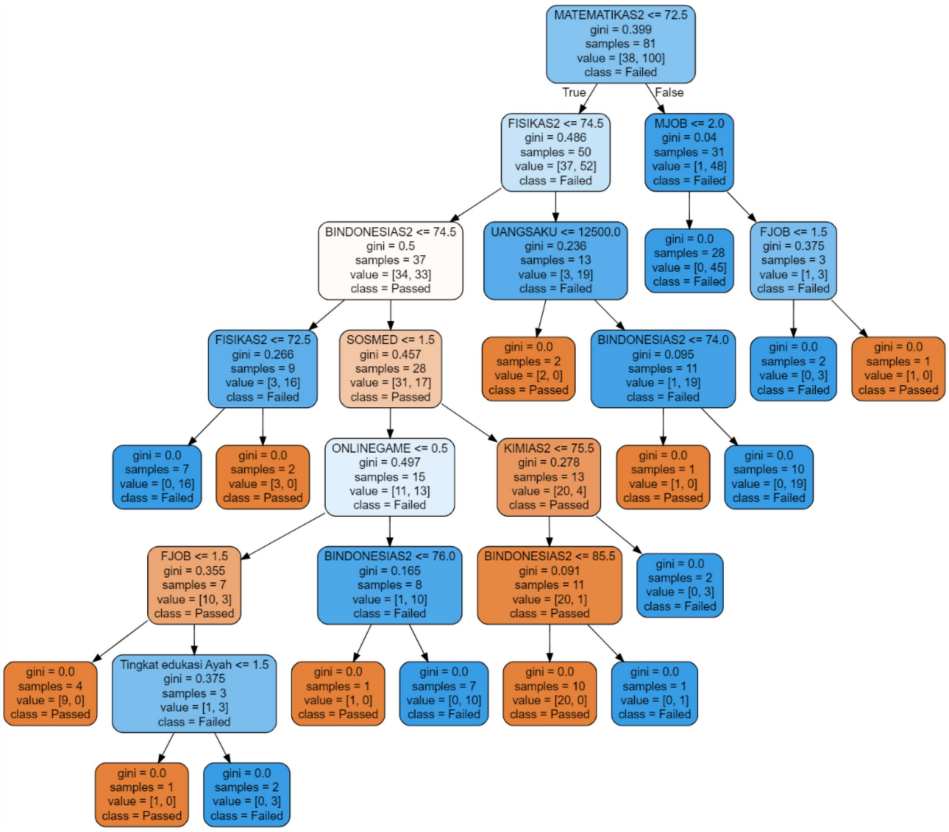
Gambar E.20 Model DT Bahasa Inggris Pohon Keputusan 10



Gambar E.21 Model DT PKn Pohon Keputusan 1

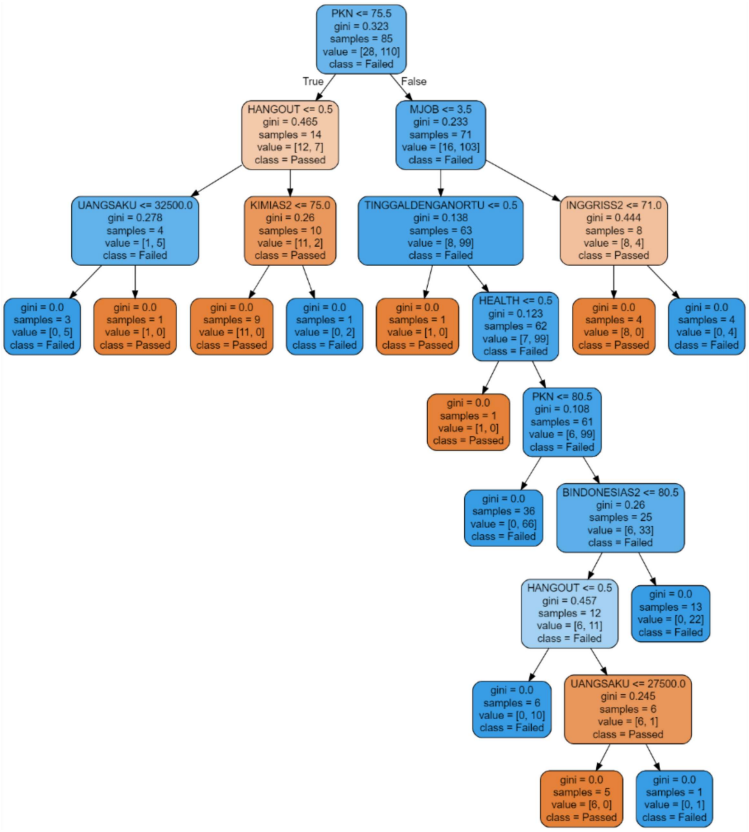


Gambar E.22 Model DT PKn Pohon Keputusan 2

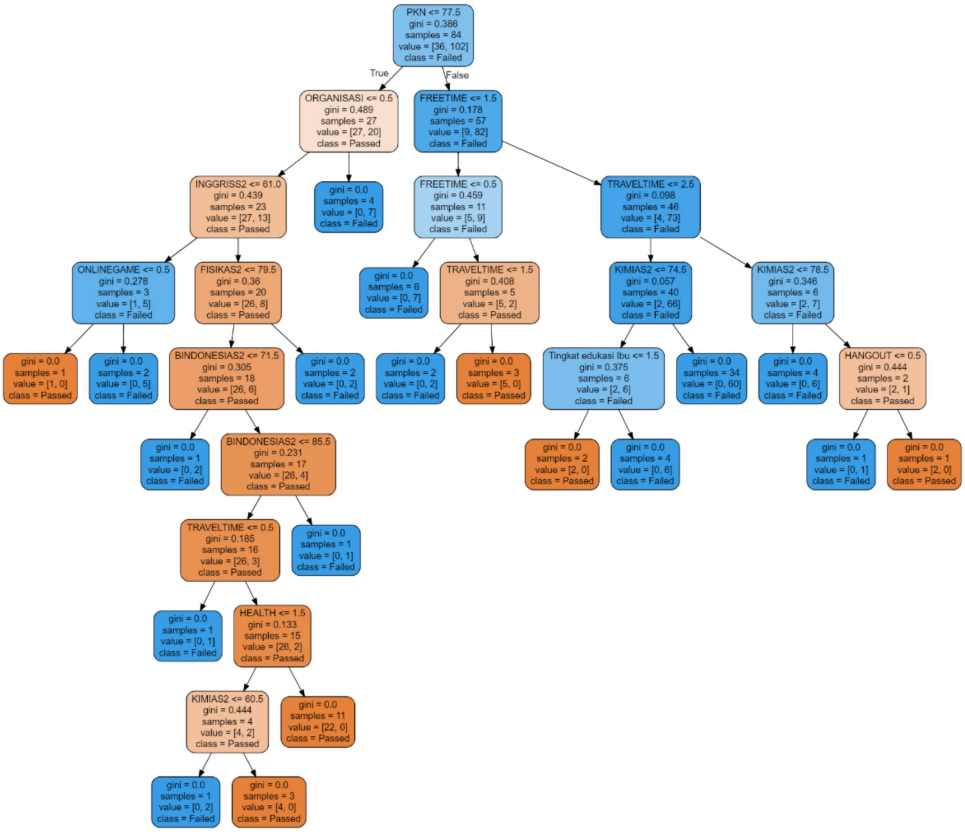


Gambar E.23 Model DT PKn Pohon Keputusan 3

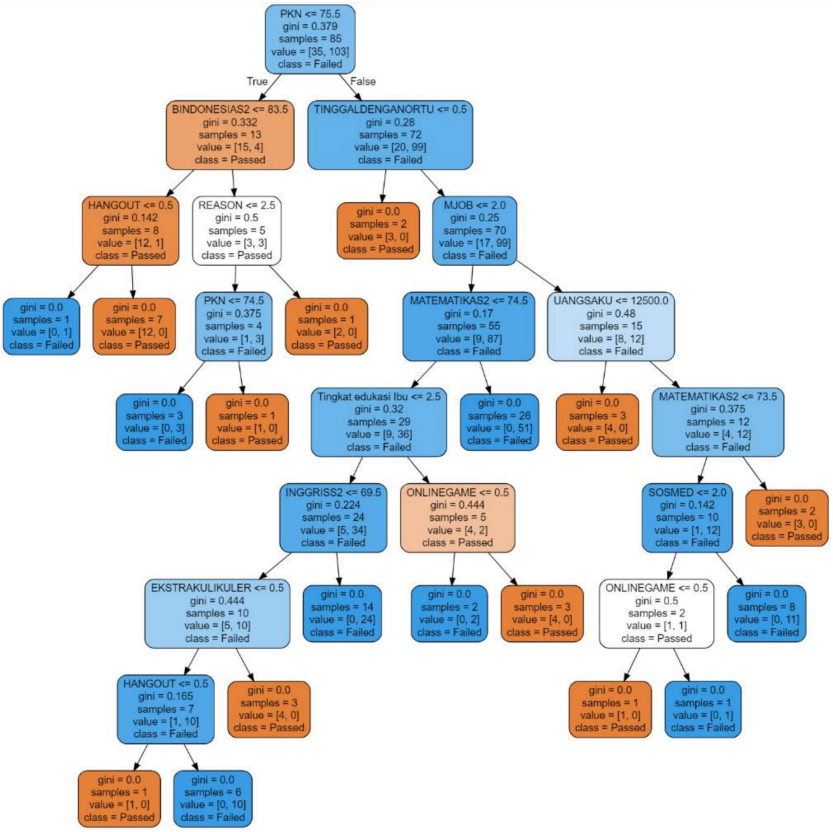
Gambar E.24 Model DT PKn Pohon Keputusan 4



Gambar E.25 Model DT PKn Pohon Keputusan 5



Gambar E.26 Model DT PKn Pohon Keputusan 6



Gambar E.27 Model DT PKn Pohon Keputusan 7

Gambar E.28 Model DT PKn Pohon Keputusan 8

Gambar E.29 Model DT PKn Pohon Keputusan 9

Gambar E.30 Model DT PKn Pohon Keputusan 10

B. Perancangan Model KNN

Perencangan model *machine learning* pada algoritma KNN untuk mata pelajaran Matematika menggunakan tahapan perancangan sebagai berikut.

1. Melakukan pembacaan data penelitian.
2. Melakukan Transformasi data dari tipe *string* ke *integer*.
3. Melakukan pembagian *training* dan testing data.
4. Membuat sistem *tuning* parameter untuk mencari parameter terbaik dari model KNN.
5. Pada algoritma KNN, hasil dari *tuning* parameter yang dilakukan menghasilkan pengaturan parameter di bawah ini.

   A. n_neighbors=5. Hasil ini menyatakan bahwa prediksi KNN berdasarkan jarak dengan 5 tetangga terdekat.

B. metric='manhattan'. Parameter ini berarti jarak antar data dihitung dengan menggunakan *Manhattan distance*. Jarak dalam *manhattan distance* dapat dituliskan sebagai:

jarak XA ke XB = (XA1 – XB1) + (XA2 – XB2) + … +(XAn – XBn)

C. Weights=uniform. Parameter ini berarti setiap data memiliki bobot yang sama.

6. Setelah didapatkan hasil parameter terbaik, selanjutnya parameter terbaik tersebut digunakan untuk membuat model *machine learning* yang dilatih dengan *training* data untuk setiap algoritma yang digunakan.
7. Hasil dari setiap model di atas kemudian digunakan untuk membuat prediksi dari testing data. Hasil dari model ini juga menghasilkan *confusion matrix*, *accuracy*, *precision*, *recall*, f1score, dan MCC.

Perencangan model *machine learning* pada algoritma KNN untuk mata pelajaran Bahasa Inggris menggunakan tahapan perancangan sebagai berikut.

1. Melakukan pembacaan data penelitian.
2. Melakukan Transformasi data dari tipe *string* ke *integer*.
3. Melakukan pembagian *training* dan testing data.
4. Membuat sistem *tuning* parameter untuk mencari parameter terbaik dari model KNN.
5. Pada algoritma KNN, hasil dari *tuning* parameter yang dilakukan menghasilkan pengaturan parameter di bawah ini.

   A. n_neighbors=6. Hasil ini menyatakan bahwa prediksi KNN berdasarkan jarak dengan 6 tetangga terdekat.

   B. metric='minkowski'. Parameter ini berarti jarak antar data dihitung dengan menggunakan *Euclidean distance*. Jarak dalam *euclidean distance* dapat dituliskan sebagai:

   jarak XA ke XB = $(XA1 – XB1)^2 + (XA2 – XB2)^2 + … +(XAn – XBn)^2$
6. Setelah didapatkan hasil parameter terbaik, selanjutnya parameter terbaik tersebut digunakan untuk membuat model *machine learning* yang dilatih dengan *training* data untuk setiap algoritma yang digunakan.

7. Hasil dari setiap model di atas kemudian digunakan untuk membuat prediksi dari testing data. Hasil dari model ini juga menghasilkan *confusion matrix*, *accuracy*, *precision*, *recall*, f1score, dan MCC.

Perencangan model *machine learning* pada algoritma KNN untuk mata pelajaran PKn menggunakan tahapan perancangan sebagai berikut.

1. Melakukan pembacaan data penelitian.
2. Melakukan Transformasi data dari tipe *string* ke *integer*.
3. Melakukan pembagian *training* dan testing data.
4. Membuat sistem *tuning* parameter untuk mencari parameter terbaik dari model KNN.
5. Pada algoritma KNN, hasil dari *tuning* parameter yang dilakukan menghasilkan pengaturan parameter di bawah ini.

   A. n_neighbors=5. Hasil ini menyatakan bahwa prediksi KNN berdasarkan jarak dengan 5 tetangga terdekat.

   B. metric='minkowski'. Parameter ini berarti jarak antar data dihitung dengan menggunakan *Euclidean distance*. Jarak dalam *euclidean distance* dapat dituliskan sebagai:

   jarak XA ke XB = $(XA1 - XB1)^2 + (XA2 - XB2)^2 + \ldots + (XAn - XBn)^2$

   D. Weights=distance. Parameter ini berarti bahwa data yang memiliki jarak terdekat memiliki bobot yang lebih besar.
6. Setelah didapatkan hasil parameter terbaik, selanjutnya parameter terbaik tersebut digunakan untuk membuat model *machine learning* yang dilatih dengan *training* data untuk setiap algoritma yang digunakan.
7. Hasil dari setiap model di atas kemudian digunakan untuk membuat prediksi dari testing data. Hasil dari model ini juga menghasilkan *confusion matrix*, *accuracy*, *precision*, *recall*, f1score, dan MCC.

C. Perancangan Model SVM

Perencangan model *machine learning* pada algoritma SVM untuk mata pelajaran Matematika menggunakan tahapan perancangan sebagai berikut.

1. Melakukan pembacaan data penelitian.

2. Melakukan transformasi data dari tipe *string* ke *integer*.
3. Melakukan pembagian *training* dan testing data.
4. Membuat sistem *tuning* parameter untuk mencari parameter terbaik dari model SVM.
5. Pada algoritma SVM, hasil dari *tuning* parameter yang dilakukan menghasilkan pengaturan parameter di bawah ini.

   A. gamma= 0.001. Ini artinya nilai koefisien gamma yang digunakan adalah sebesar 0,001. Nilai gamma ini akan mempegaruhi perhitungan pada nilai kernel.

   B. c=10. Parameter ini menentukan regularisasi yang mengendalikan seberapa keras suatu model SVM dalam hal menangani kesalahan klasifikasi pada *training* data. Nilai parameter c ini menentukan seberapa besar margin *hyperplane* yang akan terbentuk.

   C. kernel=rbf. Paramater ini berarti menentukan bahwa SVM yang dibuat menggunakan kernel rbf (*radial basis function*). Perhitungan nilai kernel rbf dapat dilakukan dengan perhitungan berikut ini.

   EXP(-gamma $|| A - B||^2$).

   Perhitungan nilai kernel inilah yang menentukan nilai untuk prediksi data. Semakin besar *euclidean distances* antar 2 titik ($|| A - B||$) maka nilai kernel yang dihasilkan semakin mendekati nilai 0. Ini artinya bahwa semakin jauh jaraknya maka kemungkinan besar data tersebut merupakan dua data yang memiliki sifat yang tidak mirip (dalam hal ini misalnya data A adalah kelas 1 dan data b adalah kelas 2).
6. Setelah didapatkan hasil parameter terbaik, selanjutnya parameter terbaik tersebut digunakan untuk membuat model *machine learning* yang dilatih dengan *training* data untuk setiap algoritma yang digunakan.
7. Hasil dari setiap model di atas kemudian digunakan untuk membuat prediksi dari testing data. Hasil dari model ini juga menghasilkan *confusion matrix*, *accuracy*, *precision*, *recall*, f1score, dan MCC.

Perencangan model *machine learning* pada algoritma SVM untuk mata pelajaran Bahasa Inggris menggunakan tahapan perancangan sebagai berikut.

1. Melakukan pembacaan data penelitian.
2. Melakukan transformasi data dari tipe *string* ke *integer*.
3. Melakukan pembagian *training* dan testing data.
4. Membuat sistem *tuning* parameter untuk mencari parameter terbaik dari model SVM.
5. Pada algoritma SVM, hasil dari *tuning* parameter yang dilakukan menghasilkan pengaturan parameter di bawah ini.

   A. gamma= 0.01. Ini artinya nilai koefisien gamma yang digunakan adalah sebesar 0,01. Nilai gamma ini akan mempegaruhi perhitungan pada nilai kernel.

   B. c=0.1. Parameter ini menentukan regularisasi yang mengendalikan seberapa keras suatu model SVM dalam hal menangani kesalahan klasifikasi pada *training* data. Nilai parameter c ini menentukan seberapa besar margin *hyperplane* yang akan terbentuk.

   C. kernel=rbf. Paramater ini berarti menentukan bahwa SVM yang dibuat menggunakan kernel rbf. Perhitungan nilai kernel rbf dapat dilakukan dengan perhitungan berikut ini.

   EXP(-gamma $|| A - B||^2$).

   Perhitungan nilai kernel inilah yang menentukan nilai untuk prediksi data. Semakin besar *euclidean distances* antar 2 titik ($|| A - B||$) maka nilai kernel yang dihasilkan semakin mendekati nilai 0. Ini artinya bahwa semakin jauh jaraknya maka kemungkinan besar data tersebut merupakan dua data yang memiliki sifat yang tidak mirip (dalam hal ini misalnya data A adalah kelas 1 dan data b adalah kelas 2).
6. Setelah didapatkan hasil parameter terbaik, selanjutnya parameter terbaik tersebut digunakan untuk membuat model *machine learning* yang dilatih dengan *training* data untuk setiap algoritma yang digunakan
7. Hasil dari setiap model di atas kemudian digunakan untuk membuat prediksi dari testing data. Hasil dari model ini juga menghasilkan *confusion matrix*, *accuracy*, *precision*, *recall*, f1score, dan MCC.

Perencangan model *machine learning* pada algoritma SVM untuk mata pelajaran PKn menggunakan tahapan perancangan sebagai berikut.

1. Melakukan pembacaan data penelitian.
2. Melakukan transformasi data dari tipe *string* ke *integer*.
3. Melakukan pembagian *training* dan testing data.
4. Membuat sistem *tuning* parameter untuk mencari parameter terbaik dari model SVM.
5. Pada algoritma SVM, hasil dari *tuning* parameter yang dilakukan menghasilkan pengaturan parameter di bawah ini.

   A. gamma= 0.001. Ini artinya nilai koefisien gamma yang digunakan adalah sebesar 0,01. Nilai gamma ini akan mempegaruhi perhitungan pada nilai kernel.

   B. c=100. Parameter ini menentukan regularisasi yang mengendalikan seberapa keras suatu model SVM dalam hal menangani kesalahan klasifikasi pada training data. Nilai parameter c ini menentukan seberapa besar margin hyperplane yang akan terbentuk.

   C. kernel=rbf. Paramater ini berarti menentukan bahwa SVM yang dibuat menggunakan kernel rbf. Perhitungan nilai kernel rbf dapat dilakukan dengan perhitungan berikut ini.

   $EXP(-gamma || A – B||^2)$.

   Perhitungan nilai kernel inilah yang menentukan nilai untuk prediksi data. Semakin besar *euclidean distances* antar 2 titik ($|| A – B||$) maka nilai kernel yang dihasilkan semakin mendekati nilai 0. Ini artinya bahwa semakin jauh jaraknya maka kemungkinan besar data tersebut merupakan dua data yang memiliki sifat yang tidak mirip (dalam hal ini misalnya data A adalah kelas 1 dan data B adalah kelas 2).
6. Setelah didapatkan hasil parameter terbaik, selanjutnya parameter terbaik tersebut digunakan untuk membuat model *machine learning* yang dilatih dengan *training* data untuk setiap algoritma yang digunakan.
7. Hasil dari setiap model di atas kemudian digunakan untuk membuat prediksi dari testing data. Hasil dari model ini juga menghasilkan *confusion matrix*, *accuracy*, *precision*, *recall*, f1score, dan MCC.

D. Perancangan Model NB

Perencangan model *machine learning* pada algoritma NB untuk mata pelajaran Matematika, Bahasa Inggris, dan PKn menggunakan tahapan perancangan sebagai berikut.

1. Melakukan pembacaan data penelitian.
2. Melakukan transformasi data dari tipe *string* ke *integer*.
3. Melakukan pembagian *training* dan testing data.
4. Membuat sistem *tuning* parameter untuk mencari parameter terbaik dari model NB.
5. Pada algoritma NB, tidak banyak parameter yang bisa dirubah karena NB bekerja berdasarkan teorema *naive bayes* dan semua variabel diasumsikan sebagai variabel independen satu dengan yang lainya. Teorema *naive bayes* dapat dilihat pada persamaan (2.1).
6. Setelah didapatkan hasil parameter terbaik, selanjutnya parameter terbaik tersebut digunakan untuk membuat model *machine learning* yang dilatih dengan *training* data untuk setiap algoritma yang digunakan.
7. Hasil dari setiap model di atas kemudian digunakan untuk membuat prediksi dari testing data. Hasil dari model ini juga menghasilkan *confusion matrix*, *accuracy*, *precision*, *recall*, f1score, dan MCC.