LAPORAN TUGAS BESAR MATA KULIAH PENAMBANGAN DATA: CLASSIFICATION

Disusun Oleh:
1301213392 - Firman Hoerulloh
1301213175 - Arrizal Aryasatya Rizqullah
1301213449 - Ariiq Afrahtama
1301213463 - Raihan Kahfi Ananta

PROGRAM STUDI S1 INFORMATIKA

FAKULTAS INFORMATIKA

UNIVERSITAS TELKOM

A. Latar Belakang

Data mining adalah proses yang menggunakan matematika, statistik, kecerdasan buatan, dan pembelajaran mesin untuk mengekstrak dan menemukan informasi terkait dan berguna dari berbagai jenis database yang sangat besar. Proses ini mencakup mengekstrak nilai tambahan dari kumpulan data yang terdiri dari pengetahuan yang sebelumnya tidak diketahui secara manual.

Klasifikasi adalah salah satu teknik pembelajaran mesin yang digunakan untuk memprediksi kelas atau label data. Bisa digunakan untuk berbagai tugas, seperti identifikasi objek, klasifikasi dokumen, dan klasifikasi suara

Metode klasifikasi asosiasi adalah pendekatan data mining terbaru yang menggabungkan metode klasifikasi dan asosiasi untuk membangun model klasifikasi. Metode ini mencapai akurasi yang lebih tinggi daripada metode klasifikasi tradisional. Paradigma klasifikasi dan pendekatan aturan asosiasi adalah dasar dari klasifikasi asosiasi. Metode ini melibatkan dua tahap proses pengolahan data. Himpunan aturan kelas (CAR) dibangun menggunakan aturan asosiasi yang sering digunakan selama fase pembentukan aturan. Setelah itu, data yang belum memiliki label diklasifikasikan atau diprediksi dengan menggunakan model klasifikasi selama fase pembangunan pengklasifikasi. CBA, CMAR, MCAR, dan L3G adalah beberapa algoritma yang digunakan untuk klasifikasi asosiasi.

B. Ringkasan Metode

Pada laporan Project-Based Assignment CLO-3 metode yang digunakan yaitu algoritma Associative Classification. Associative Classification adalah pendekatan dalam data mining yang menggabungkan teknik asosiasi dan klasifikasi untuk membangun model klasifikasi. Tujuan utama algoritma ini adalah untuk menemukan aturan asosiasi yang kuat antara atribut-atribut dalam data dan menggunakannya untuk melakukan klasifikasi.

Dalam Associative Classification terdiri dari beberapa langkah utama, diantaranya:

1. **Pembersihan Data**: Langkah pertama adalah pembersihan data untuk menghilangkan noise, menangani nilai yang hilang, dan memastikan konsistensi data. Data yang bersih sangat penting karena kualitas data sangat mempengaruhi akurasi model yang akan dibangun. Proses ini melibatkan berbagai teknik seperti imputasi data yang hilang, normalisasi, dan transformasi data.
2. **Ekstraksi Aturan Asosiasi**: Setelah data dibersihkan, langkah berikutnya adalah mengekstraksi aturan asosiasi dari data menggunakan algoritma seperti Apriori atau FP-Growth. Aturan asosiasi adalah pola yang ditemukan dalam data yang menunjukkan hubungan antara atribut yang berbeda. Misalnya, dalam dataset pembelian, aturan bisa berbentuk "jika seseorang membeli A,

mereka juga cenderung membeli B". Proses ini menghasilkan sejumlah aturan yang memiliki confidence dan support tertentu.

3. **Pemilihan Aturan Klasifikasi**: Dari aturan asosiasi yang telah diekstraksi, hanya aturan yang memenuhi threshold tertentu (misalnya, minimum support dan confidence) yang dipilih sebagai aturan klasifikasi. Pemilihan ini penting untuk memastikan hanya aturan yang paling relevan dan kuat yang digunakan dalam model klasifikasi. Proses ini juga bisa melibatkan penentuan kelas target yang paling mungkin berdasarkan aturan yang dipilih.
4. **Pembangunan Model Klasifikasi**: Dengan aturan klasifikasi yang telah dipilih, model klasifikasi kemudian dibangun. Model ini menggunakan aturan untuk memprediksi kelas dari data baru. Setiap aturan memiliki bentuk "jika kondisi, maka kelas" dan digunakan untuk mengklasifikasikan instance baru berdasarkan atribut mereka. Model ini dioptimalkan untuk memaksimalkan akurasi prediksi.
5. **Evaluasi Model**: Langkah terakhir adalah evaluasi model untuk menilai kinerjanya. Ini melibatkan penggunaan metrik seperti akurasi, presisi, recall, dan F-measure untuk menilai seberapa baik model dapat mengklasifikasikan data baru. Evaluasi ini sering dilakukan menggunakan teknik validasi silang atau dengan memisahkan dataset menjadi set pelatihan dan set pengujian. Hasil evaluasi digunakan untuk mengidentifikasi kekuatan dan kelemahan model, serta untuk melakukan tuning lebih lanjut jika diperlukan.

C. Hasil dan Analisis

Preprocessing

```
import pandas as pd
import numpy as np
import gdown
import re
from google.colab import files
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, precision_score, recall_score, f1_score
import seaborn as sns
import matplotlib.pyplot as plt
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules
from collections import Counter
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer, PorterStemmer
from nltk.tokenize import word_tokenize
import spacy
import string
from joblib import Parallel, delayed
from scipy.sparse import csr_matrix
import pyarc
from pyarc import TransactionDB, CBA
from pyarc.algorithms import (
    top_rules,
    createCARs,
    M1Algorithm,
    M2Algorithm
)
```

```
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
```

Mengunduh *'stopwords', 'punkt', 'wordnet'* dari library nltk, untuk mengolah dan menganalisis kalimat atau kata.

```
url = f'https://drive.google.com/uc?id=1m3dAb7jy4b1-Vh9v-ox8d2m6NLD1moj4'
output = '20newsgroups.zip'
gdown.download(url, output, quiet=False)
```

Download Dataset 20newsgroups

```
[ ] # Load the CSV file
    df_list = pd.read_csv('/content/20newsgroups/list.csv')

    # Display the first few rows of the dataframe
    print("First few rows of list.csv:")
    print(df_list.head())

    # Display the unique categories in the CSV file
    unique_categories_csv = df_list['newsgroup'].drop_duplicates().sort_values().to_list()
    print("Number of categories: ", len(unique_categories_csv))
    print("Unique categories in list.csv:", unique_categories_csv)
```

membaca data pada dataset dan menampilkan memvalidasi isi data serta menganalisis kategori yang tersedia dalam kolom newsgroup.

```
categories_20newsgroup = [
    'alt.atheism.txt', 'comp.graphics.txt', 'comp.os.ms-windows.misc.txt',
    'comp.sys.ibm.pc.hardware.txt', 'comp.sys.mac.hardware.txt', 'comp.windows.x.txt',
    'misc.forsale.txt', 'rec.autos.txt', 'rec.motorcycles.txt',
    'rec.sport.baseball.txt', 'rec.sport.hockey.txt', 'sci.crypt.txt',
    'sci.electronics.txt', 'sci.med.txt', 'sci.space.txt',
    'soc.religion.christian.txt', 'talk.politics.guns.txt', 'talk.politics.mideast.txt',
    'talk.politics.misc.txt', 'talk.religion.misc.txt'
]

# Define the directory containing the files
directory = '/content/20newsgroups'

# Initialize an empty list to hold the data
data = []

# Process each file in the category list
for newsgroup_file in categories_20newsgroup:
    newsgroup_name = newsgroup_file.replace('.txt', '')
    file_path = os.path.join(directory, newsgroup_file)
    try:
        with open(file_path, 'r', errors='ignore') as file:
            document_id = 1
            document_text = ''
            for line in file:
                # Check for document separator or end of file
                if line.strip() == '' and document_text:
                    data.append([document_id, newsgroup_name, document_text.strip()])
                    document_id += 1
                    document_text = ''
```

```
                else:
                    document_text += line
            # Add last document if file doesn't end with a blank line
            if document_text:
                data.append([document_id, newsgroup_name, document_text.strip()])
    except FileNotFoundError:
        print(f"File not found: {file_path}")

# Display the first few rows of the data
for row in data[:5]:
    print(row)
```

memilih dokumen dari beberapa kategori spesifik, membaca teks dari file yang relevan, mengatur data dalam format yang terstruktur (dataframe), dan memastikan bahwa kategori yang dipilih telah terwakili dengan benar dalam data yang dihasilkan

```python
# Create a DataFrame
df = pd.DataFrame(data, columns=['Id', 'Newsgroup', 'Text'])

# Display the first few rows of the DataFrame
print("The first few rows of the dataframe :")
print(df.head())

# Display the last few rows of the dataframe
print("\nThe last few rows of the dataframe :")
print(df.tail())

# Number of unique categories
num_categories = df['Newsgroup'].nunique()

# Unique categories in the DataFrame
unique_categories = df['Newsgroup'].unique()

print("\nNumber of categories: ", num_categories)
print("Unique categories in the DataFrame: ", unique_categories)
```

Kode ini membantu untuk memahami struktur dan isi dari DataFrame yang dibuat, termasuk menampilkan beberapa baris pertama dan terakhir serta menganalisis jumlah dan daftar kategori unik yang ada.

```python
# Select 6 specific categories
selected_newsgroups = [
    'alt.atheism.txt', 'comp.graphics.txt',
    'comp.sys.ibm.pc.hardware.txt', 'comp.windows.x.txt',
    'rec.motorcycles.txt', 'rec.sport.baseball.txt'
]
print("Selected newsgroups:", selected_newsgroups)

# Initialize an empty list to hold the data
data = []

# Read each file and extract the documents
for newsgroup_file in selected_newsgroups:
    newsgroup_name = newsgroup_file.replace('.txt', '')
    file_path = os.path.join(directory, newsgroup_file)
    try:
        with open(file_path, 'r', errors='ignore') as file:
            document_id = 1
            document_text = ''
            for line in file:
                # Check for document separator or end of file
                if line.strip() == '' and document_text:
                    data.append([document_id, newsgroup_name, document_text.strip()])
                    document_id += 1
                    document_text = ''
                else:
                    document_text += line
            # Add last document if file doesn't end with a blank line
            if document_text:
                data.append([document_id, newsgroup_name, document_text.strip()])
    except FileNotFoundError:
```

```python
            print(f"File not found: {file_path}")

# Create a DataFrame
df = pd.DataFrame(data, columns=['Id', 'Newsgroup', 'Text'])

# Number of unique categories
num_categories = df['Newsgroup'].nunique()

# Unique categories in the DataFrame
unique_categories = df['Newsgroup'].unique()

# Display results
print("Number of unique categories:", num_categories)
print("Unique categories in the DataFrame:", unique_categories)
```

memproses dan mengorganisir data dari kategori-kategori tertentu dalam sebuah dataset teks. Pertama, kategori-kategori spesifik dari dataset ditetapkan, dan file-file teks yang sesuai dengan kategori-kategori tersebut dibaca. Setelah itu, setiap dokumen dalam setiap kategori diproses secara individu, dan teks dari dokumen-dokumen tersebut diekstrak dengan memperhatikan batas antara satu dokumen dengan dokumen lainnya. Data hasilnya kemudian disusun dalam sebuah DataFrame, yang memungkinkan untuk analisis lebih lanjut. Setelah pemrosesan, informasi seperti jumlah kategori unik dan daftar kategori-kategori yang ada dipresentasikan. Keseluruhan proses ini bertujuan untuk menyiapkan data untuk analisis lebih lanjut, seperti pemrosesan teks dan pembangunan model klasifikasi.

```python
# Select 6 specific categories
selected_newsgroups = [
    'alt.atheism.txt', 'comp.graphics.txt',
    'comp.sys.ibm.pc.hardware.txt', 'comp.windows.x.txt',
    'rec.motorcycles.txt', 'rec.sport.baseball.txt'
]
print("Selected newsgroups:", selected_newsgroups)

# Initialize an empty list to hold the data
data = []

# Read each file and extract the documents
for newsgroup_file in selected_newsgroups:
    newsgroup_name = newsgroup_file.replace('.txt', '')
    file_path = os.path.join(directory, newsgroup_file)
    try:
        with open(file_path, 'r', errors='ignore') as file:
            document_id = 1
            document_text = ''
            for line in file:
                # Check for document separator or end of file
                if line.strip() == '' and document_text:
                    data.append([document_id, newsgroup_name, document_text.strip()])
                    document_id += 1
                    document_text = ''
                else:
                    document_text += line
            # Add last document if file doesn't end with a blank line
            if document_text:
                data.append([document_id, newsgroup_name, document_text.strip()])
    except FileNotFoundError:
        print(f"File not found: {file_path}")

# Create a DataFrame
df = pd.DataFrame(data, columns=['Id', 'Newsgroup', 'Text'])

# Number of unique categories
num_categories = df['Newsgroup'].nunique()

# Unique categories in the DataFrame
unique_categories = df['Newsgroup'].unique()

# Display results
print("Number of unique categories:", num_categories)
print("Unique categories in the DataFrame:", unique_categories)
```

```
Selected newsgroups: ['alt.atheism.txt', 'comp.graphics.txt', 'comp.sys.ibm.pc.hardware.txt', 'comp.windows.x.txt', 'rec.motorcycles.txt', 'rec.sport.baseball.txt']
Number of unique categories: 6
Unique categories in the DataFrame: ['alt.atheism' 'comp.graphics' 'comp.sys.ibm.pc.hardware' 'comp.windows.x'
 'rec.motorcycles' 'rec.sport.baseball']
```

Dalam kode tersebut, 6 kategori spesifik dipilih dari dataset teks untuk diproses. Setiap file teks dari kategori-kategori tersebut dibaca, dan dokumen-dokumen di dalamnya diekstrak. Dokumen-dokumen tersebut kemudian disusun ke dalam sebuah DataFrame, yang berisi ID dokumen, nama kategori, dan teks dokumen. Jumlah kategori unik dan daftar kategori-kategori yang ada juga dihitung dan dipresentasikan. Langkah-langkah ini

membantu dalam menyiapkan data untuk analisis lebih lanjut, seperti pemrosesan teks atau pembangunan model klasifikasi.

```
# Mengelompokkan dataframe berdasarkan nilai kolom 'Newsgroup'
grouped_df = df.groupby('Newsgroup')

# Membuat list untuk menyimpan DataFrame hasil sampling
sampled_dfs = []

# Mengambil 1000 baris pertama dari setiap kelompok dan menyimpannya dalam list
for group_name, group_df in grouped_df:
    sampled_dfs.append(group_df.head(2000))

# Menggabungkan DataFrames dari list
sampled_df = pd.concat(sampled_dfs, ignore_index=True)

# Menampilkan distribusi kategori pada dataframe baru
print("Category Distribution in Sampled DataFrame:")
print(sampled_df['Newsgroup'].value_counts())
```

Mengelompokkan dataframe sesuai dengan NewsGroup, disini kami mengambil 2000 data per kategori

```
def remove_punctuation_without_space(input_string):

  punctuation = string.punctuation

  result = ""
  for char in input_string:
    if char in punctuation and char != ':':
      continue  # Skip punctuation characters except colon
    result += char

  return result
```

Menghapus tanda semua tanda baca kecuali ‘:’, karena tanda baca ‘:’ menentukan arti dari sebuah kalimat, contoh dalam penulisan waktu.

```python
# Muat data ke dalam DataFrame
df = pd.DataFrame(sampled_df)

# Periksa kolom dalam DataFrame
print("Kolom dalam DataFrame:", df.columns)

# Terapkan fungsi ke kolom 'Text'
if 'Text' in df.columns:
    df['Text_Cleaned'] = df['Text'].apply(remove_punctuation_without_space)
    # Tampilkan hasil
    print(df)
else:
    print("Kolom 'Text' tidak ditemukan dalam DataFrame.")
```

Memanfaatkan function remove_punctuation_without_space pada dataframe dan menyimpannya pada kolom Text_Cleaned.

```python
def clean_text(text):
    # remove '\n'
    text = re.sub(r'\n', ' ', text)
    # remove number
    text = re.sub(r"\d+", "", text)
    # tokenize the text
    words = word_tokenize(text)
    # convert to lowercase
    words = [word.lower() for word in words]
    # remove punctuation
    words = [remove_punctuation_without_space(word) for word in words]
    # remove empty string
    words = [word for word in words if word != ""]
    # remove stopwords
    stop_words = set(stopwords.words('english'))
    words = [word for word in words if word not in stop_words]
    # remove words that have length <= 2
    words = [word for word in words if len(word) > 2]
    # create a lemmatizer object
    lemmatizer = WordNetLemmatizer()
    words = [lemmatizer.lemmatize(word) for word in words]
    return words
```

Kode diatas berfungsi untuk membersihkan teks mentah melalui beberapa langkah, termasuk penghapusan karakter khusus, tokenisasi, normalisasi, dan penghapusan kata-kata umum (stopwords).

```python
# function to create vocabulary that appear in the dataset
def create_vocabulary(token_lists):
    all_tokens = [token for tokens in token_lists for token in tokens]
    return list(set(all_tokens))

def create_binary_features(df, column_name, min_occurrences=100):
    vocabulary = create_vocabulary(df[column_name])

    # create a dictionary to store binary features
    binary_features_dict = {word: [] for word in vocabulary}

    # populate the dictionary
    for tokens in df[column_name]:
        for word in vocabulary:
            binary_features_dict[word].append(1 if word in tokens else 0)

    # create a new DataFrame with binary features
    binary_features_df = pd.DataFrame(binary_features_dict)

    # prune columns based on the minimum occurrences
    columns_to_prune = [col for col in binary_features_df.columns if binary_features_df[col].sum() < min_occurrences]

    # drop the columns with low frequency of occurrences.
    binary_features_df = binary_features_df.drop(columns=columns_to_prune)

    # concatenate the new DataFrame with the original DataFrame
    df = pd.concat([df, binary_features_df], axis=1)

    return df
```

Pada function create_vocabulary membuat daftar kosakata unik pada setiap kalimat, kemudian mengelompokkan embuat fitur biner berdasarkan kolom teks dalam DataFrame, dan menghapus fitur yang muncul kurang dari min_occurrences

Pembuatan Model Baseline

```python
txns_train = TransactionDB.from_DataFrame(df_train, target='Newsgroup')
txns_test = TransactionDB.from_DataFrame(df_test, target='Newsgroup')
```

Kode di atas mengonversi DataFrame df_train dan df_test menjadi objek TransactionDB yang diperlukan untuk analisis asosiasi. Data diubah ke dalam format yang sesuai, dengan menentukan kolom target sebagai 'Newsgroup'. Ini memungkinkan penggunaan data ini dalam analisis asosiasi atau algoritma lain yang memerlukan struktur transaksi.

```python
# get the best association rules
rules = top_rules(txns_train.string_representation, appearance=txns_train.appeardict, target_rule_count=100, init_support=0.2, init_conf=0.7)

# convert them to class association rules
cars = createCARs(rules)

classifier = M1Algorithm(cars, txns_train).build()

y_pred = classifier.predict_all(txns_test)
```

Kode ini mengimplementasikan proses untuk menemukan aturan asosiasi terbaik dari data pelatihan, mengonversi aturan tersebut menjadi Class Association Rules, membangun classifier berdasarkan aturan tersebut menggunakan Algoritma M1, dan akhirnya menggunakan classifier untuk memprediksi label kelas untuk data uji.

Analisa Hasil

```
# Evaluate the classification results
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

print("Classification Report:")
print(classification_report(y_test, y_pred, zero_division=0))
```

```
Accuracy: 0.5241666666666667
Classification Report:
                          precision    recall  f1-score   support

             alt.atheism       0.58      0.56      0.57       402
           comp.graphics       0.53      0.26      0.35       426
comp.sys.ibm.pc.hardware       0.63      0.52      0.57       408
          comp.windows.x       0.56      0.57      0.56       381
        rec.motorcycles       0.35      0.66      0.46       372
     rec.sport.baseball       0.65      0.59      0.62       411

                accuracy                           0.52      2400
               macro avg       0.55      0.53      0.52      2400
            weighted avg       0.55      0.52      0.52      2400
```

Kode di atas mengevaluasi hasil klasifikasi dengan menghitung akurasi menggunakan metode accuracy_score dari scikit-learn, didapat Akurasi Model 0.5241666666666667. Selain itu didapati juga nila precision, recall, f1-score, dan support dari masing-masing kategori.

```
# Calculate evaluation metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='macro', zero_division=0)
recall = recall_score(y_test, y_pred, average='macro', zero_division=0)
f1 = f1_score(y_test, y_pred, average='macro', zero_division=0)

print("Evaluation metrics for Associative Classification:")
print(f"Accuracy : {accuracy:.2f}")
print(f"Precision : {precision:.2f}")
print(f"Recall : {recall:.2f}")
print(f"F1 Score : {f1:.2f}")
```

```
Evaluation metrics for Associative Classification:
Accuracy : 0.52
Precision : 0.55
Recall : 0.53
F1 Score : 0.52
```

Menghitung nilai evaluasi metriks, ringkasan hasil tersebut memberikan gambaran tentang seberapa baik model klasifikasi yang menggunakan aturan asosiasi dalam memprediksi kelas

dalam data uji. Semakin tinggi nilai akurasi, presisi, recall, dan F1-score, semakin baik kinerja model.

Evaluasi

Pada tahap Evaluasi, yang kita ubah adalah init_support yang semula 0.2 menjadi 0.1, dan merubah init_conf yang semula 0.7 menjadi 0.5.

Menurunkan nilai init_support dapat menghasilkan aturan asosiasi yang lebih banyak, dan dapat meningkatkan kemungkinan menemukan pola atau asosiasi yang lebih halus atau lebih jarang terjadi dalam data.

Menurunkan nilai init_conf akan memperlonggar kriteria untuk kepercayaan aturan yang diterima. Ini berarti aturan dengan tingkat kepercayaan yang lebih rendah akan disertakan dalam kumpulan aturan yang dihasilkan. Hal ini dapat menyebabkan model untuk lebih toleran terhadap asosiasi yang tidak sekuat atau tidak sejelas sebelumnya.

```
# get the best association rules / init_support=0.1 and int_conf=0.5
rules2 = top_rules(txns_train.string_representation, appearance=txns_train.appeardict, target_rule_count=100, init_support=0.1, init_conf=0.5)

# convert them to class association rules
cars = createCARs(rules2)

classifier = M1Algorithm(cars, txns_train).build()

y_pred = classifier.predict_all(txns_test)
```

```
Accuracy: 0.5283333333333333
Classification Report:
                          precision    recall  f1-score   support

             alt.atheism       0.58      0.55      0.57       402
           comp.graphics       0.57      0.28      0.38       426
comp.sys.ibm.pc.hardware       0.64      0.52      0.58       408
          comp.windows.x       0.58      0.59      0.58       381
        rec.motorcycles       0.35      0.65      0.45       372
     rec.sport.baseball       0.64      0.59      0.61       411

               accuracy                           0.53      2400
              macro avg       0.56      0.53      0.53      2400
           weighted avg       0.56      0.53      0.53      2400
```

```
Evaluation metrics for Associative Classification:
Accuracy : 0.53
Precision : 0.56
Recall : 0.53
F1 Score : 0.53
```

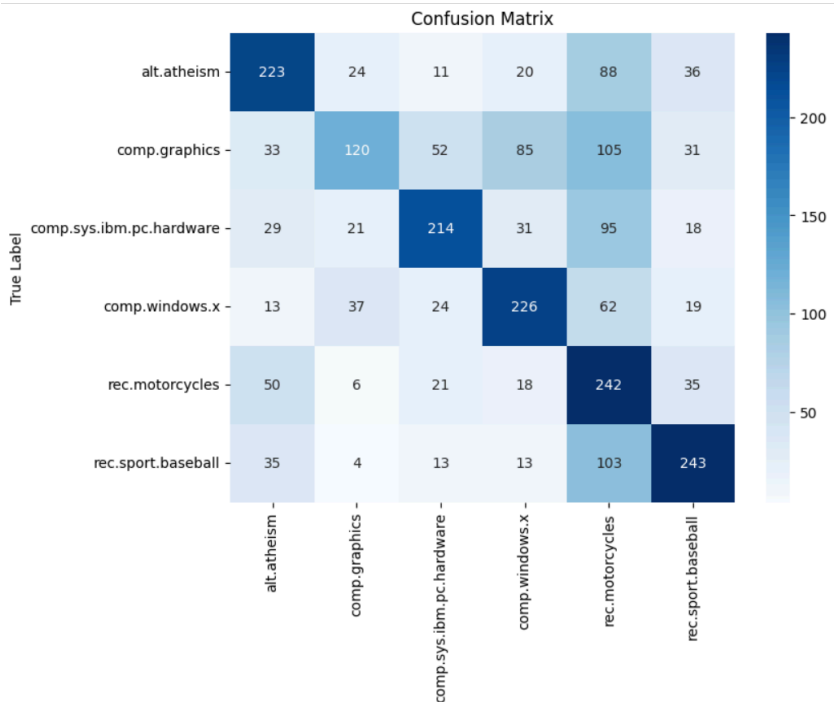
Didapati hasil seperti diatas, tidak terdapat banyak perubahan dari pemrosesan awal. hanya terdapat selisih 0.1 atau 1% lebih baik pada percobaan evaluasi.

```
# Buat confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Tampilkan confusion matrix
print("Confusion Matrix:")
print(conf_matrix)
```

```
# Nama kelas
class_names = ['alt.atheism', 'comp.graphics', 'comp.sys.ibm.pc.hardware', 'comp.windows.x', 'rec.motorcycles', 'rec.sport.baseball']

# Visualisasi confusion matrix sebagai heatmap dengan nama kelas
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=class_names, yticklabels=class_names)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()
```

Kemudian membuat confussion matrix dari hasil prediksi model klasifikasi dibandingkan dengan label sebenarnya dari dataset uji.

Didapati Confussion Matrix seperti dibawah,



semakin besar nilai di luar diagonal utama, semakin buruk kinerja model karena ini menunjukkan bahwa model sering membuat kesalahan dalam mengklasifikasikan sampel.

Kesimpulan

Dengan mengubah parameter init_support dan init_conf dalam pencarian aturan asosiasi, terjadi peningkatan kecil dalam akurasi klasifikasi dari sekitar 52% menjadi sekitar 53%.

Meskipun peningkatan ini tidak terlalu signifikan, beberapa metrik dalam laporan klasifikasi mengalami perubahan, menunjukkan responsibilitas model terhadap perubahan parameter.