IMPLEMENTASI MULTIPROCESSING PADA KLASIFIKASI DATA ANEMIA

Implementation of Multiprocessing on Anemia Data Classification

Resti Ika Pertiwi[a], Siti Khoirun Nisak[b], Sari Mumtahanah Sani[c], Nanda Ariba Althaf[d], Sofia Yuliana Manullang[e]

Universitas Jember[abcde]

212410103005@gmail.unej.ac.id[a], 212410103010@gmail.unej.ac.id[b], 212410103023@gmail.unej.ac.id[c], 212410103032@gmail.unej.ac.id[d], 212410103069@gmail.unej.ac.id[e]

Jalan Kalimantan Tegalboto No. 37, Krajan Timur, Sumbersari, Kecamatan Sumbersari, Kabupaten Jember, Jawa Timur 68121

INFO ARTIKEL

Kata kunci:
Anemia
Random Forest
Multiprosesor
Klasifikasi

Keywords:
Anemia
Random Forest
Multiprocessing
Classification

Style APA dalam mensitasi artikel ini:
Untoro, M. C., Anggraini, L., Andini, M., Retnosari, H., &

ABSTRAK

Anemia merupakan kondisi kesehatan yang ditandai dengan rendahnya kadar hemoglobin di dalam darah. Klasifikasi dari data anemia menjadi penting untuk proses pengolahan data dan diagnosis dini. Teknik penyajian data klasifikasi yang diolah dengan konsep multiprocessing dalam bahasa pemrograman Python. Penelitian ini dilakukan untuk meningkatkan efisiensi dan kecepatan klasifikasi data anemia. Hasil penelitian yang dilakukan menggunakan dataset yang bersumber dari situs resmi dan terpercaya. Dengan adanya penelitian ini diharapkan pembaca dapat memahami cara kerja multiprocessing dan mengetahui kegunaan metode tersebut dalam pengelolaan data. Dalam kasus ini data yang dolah adalah dataset anemia yang dimana ingin memprediksi ciri-ciri pengidap anemia. Hal tersebut menunjukkan bahwa dengan hemoglobin yang rendah, orang dominan mengidap anemia, tetapi tidak semua orang yang memiliki hemoglobin di bawah normal akan mengidap anemia dikarenakan adanya faktor lainnya.

ABSTRACT

Anemia is a health condition characterized by low hemoglobin levels in the blood. Classification of anemia data is important for data processing and early diagnosis. The classification data presentation technique is processed

Nasrulloh, M. A. (2021). Penerapan metode k-means clustering data COVID-19 di Provinsi Jakarta. Teknologi: Jurnal Ilmiah Sistem Informasi, 11(2), 59-68.

with the concept of multiprocessing in the Python programming language. This research was conducted to improve the efficiency and speed of anemia data classification. The results of the research conducted using datasets sourced from official and trusted sites. With this research, it is hoped that readers can understand how multiprocessing works and know the usefulness of these methods in data management. In this case, the data processed is an anemia dataset which wants to predict the characteristics of anemia sufferers. It shows that with low hemoglobin, the dominant person has anemia, but not all people who have hemoglobin below normal will have anemia due to other factors.

1. Pendahuluan

Anemia merupakan kondisi kesehatan yang ditandai dengan rendahnya kadar hemoglobin di dalam darah. Hemoglobin merupakan protein yang terdapat di dalam sel darah merah yang mempunyai peran penting untuk mengangkut karbon dioksida dari paru-paru ke seluruh tubuh. Besaran kadar hemoglobin merepresentasikan kemampuan dari darah untuk membawa oksigen. Kadar hemoglobin normal untuk pria dewasa sekitar 13,2 - 17,5 g/dL, wanita dewasa 11,6 - 15,5 g/dL, dan jika di bawah kadar normal dapat diidentifikasikan kemungkinan terkena anemia.

Di dalam darah terdapat kadar hemoglobin yang memiliki kemampuan untuk membawa oksigen. Dalam keadaan normal, pria dewasa memiliki kadar hemoglobin sekitar 13,2-17,5 gram per desiliter (g/dL) darah, jika wanita dewasa sekitar 11,6-15,5 g/dL. Anemia atau kondisi lainnya dapat dilihat atau diindikasi melalui kadar hemoglobin yang rendah.

Protein yang ada di dalam sel darah merah atau eritrosit merupakan pengertian dari hemoglobin. Hemoglobin memiliki peran penting, yaitu mengangkut oksigen dari paru-paru ke seluruh tubuh. Selain itu, hemoglobin memiliki peran dalam pengangkutan karbondioksida dari jaringan tubuh lalu kembali ke paru-paru yang akan dikeluarkan.

Penelitian ini memiliki tujuan untuk melakukan klasifikasi terhadap dataset anemia dengan menggunakan dataset yang telah disajikan pada portal https://www.kaggle.com/datasets/biswaranjanrao/anemia-dataset. Dataset tersebut memiliki atribut berupa jenis kelamin, hemoglobin, MCH, MCHC, MCV, dan hasil. Tentunya dataset ini dapat digunakan untuk dapat memprediksi kemungkinan seseorang menderita anemia atau tidak.

Di dalam data mining terdapat beberapa jenis algoritma, salah satunya yaitu klasifikasi. “Klasifikasi merupakan jenis analisis data yang berguna untuk menentukan kelas label dari suatu sampel yang akan diklasifikasi.

Klasifikasi memiliki tujuan untuk meningkatkan kemampuan hasil dari data yang diperoleh. Di mana ketika nilai akurasi semakin tinggi, maka kemampuan algoritma tersebut semakin baik dalam mengklasifikasikan data." (Hendrian,S. 2018)

Klasifikasi terdiri dari beberapa algoritma yang sering ditemui dan digunakan, yaitu Support Vector Machine (SVM), Decision Tree, dan Naive Bayes. Seperti tujuannya, algoritma klasifikasi memiliki nilai algoritma yang menjadi salah satu poin yang penting dalam memperhitungkan ketepatan suatu algoritma klasifikasi itu sendiri.

2. State of The Art

Penelitian yang berfokus pada mengimplementasikan *multiprocessing* pada klasifikasi *Anemia Dataset* belum ada. Penelitian dengan judul Pembangunan Aplikasi Identifikasi Kesalahan Ketik Dokumen Berbahasa Indonesia Menggunakan Algoritma *Jaro-Winkler Distance.* Diambil dari Jurnal Informatika, diteliti oleh Grelly Lucia Yovellia Londo, Yohanes Sigit Purnomo W.P., dan Martinus Maslim pada tahun 2020 di Yogyakarta. Penelitian ini menggunakan *parallel processing* untuk mempercepat waktu eksekusi model dikarenakan banyaknya dataset dan data masukan yang digunakan. Hasil yang didapatkan hanya dapat mengecek sejumlah 61.309 daftar kata bahasa Indonesia dari KBBI menggunakan algoritma *Jaro-Winkler distance* dan untuk mempercepat waktu eksekusinya menggunakan *parallel processing*. (Yovellia Londo et al., 2020, 27)

Penelitian dengan judul Klasifikasi Topik Twitter menggunakan Metode Random Forest dan Fitur Ekspansi Word2Vec. Diambil dari jurnal *e-Proceeding of Engineering,* diteliti oleh Rafly Ghazali Ramli dan Yuliant Sibaroni pada tahun 2022 di Bandung. Penelitian ini menggunakan metode Random Forest untuk pengklasifikasian data tweet. Penggunaan metode ini karena mampu menjaga apabila terdapat ketidak seimbangan data pada kelas yang berbeda, khususnya kumpulan data yang sangat banyak. (Ramli & Sibaroni, 2022, 90)

Penelitian dengan judul *Implementation of Multiprocessing and Multithreading for End Node Middleware Control on Internet of Things Devices.* Diambil dari Jurnal Ilmiah Bidang Teknologi Informasi dan Komunikasi, diteliti oleh Iwan Kurnianto Wibowo, Adnan Rachmat Anom Besari, dan Muh. Rifqi Rizqullah pada tahun 2021 di Surabaya. Penelitian ini menggunakan multiprocessing, perangkat IoT dapat menjalankan beberapa proses secara bersamaan. Hal ini memungkinkan pengolahan data yang lebih cepat dan responsif, serta memungkinkan perangkat untuk menangani tugas yang lebih kompleks. Dalam penelitian ini, implementasi multiproses dan multithreading berhasil meningkatkan pengendalian middleware pada perangkat IoT. Hal ini berdampak positif terhadap kemampuan perangkat untuk mengelola data, memproses permintaan, dan menjalankan tugas-tugas yang diperlukan. (Wibowo et al., 2021, 60)

Penelitian dengan judul *Energy Efficient Multiprocessing Solo Mining Algorithms for Public Blockchain Systems.* Diambil dari Jurnal Scientific Programming, diteliti oleh Zeeshan Raza, Irfan ul Haq, Muhammad Muneeb, dan Omair Shafiq pada tahun 2021 di Canada. Penelitian ini menggunakan pemrosesan paralel untuk menyelesaikan berbagai masalah kompleks dan juga dapat digunakan dalam penambangan blockchain untuk membuatnya lebih cepat. Penambangan PoW adalah protokol penambangan blockchain yang terkenal dan digunakan dalam banyak sistem berbasis blockchain karena sifatnya yang kompleks, ia menghadapi masalah waktu dan konsumsi energi. Untuk meningkatkan masalah ini, peneliti membangun dua algoritma penambangan PoW solo berbasis pemrosesan paralel, dimana beberapa proses menyelesaikan teka-teki matematika kompleks pada nilai nonce yang berbeda. (Raza et al., 2021, 10)

3. Metode Penelitian

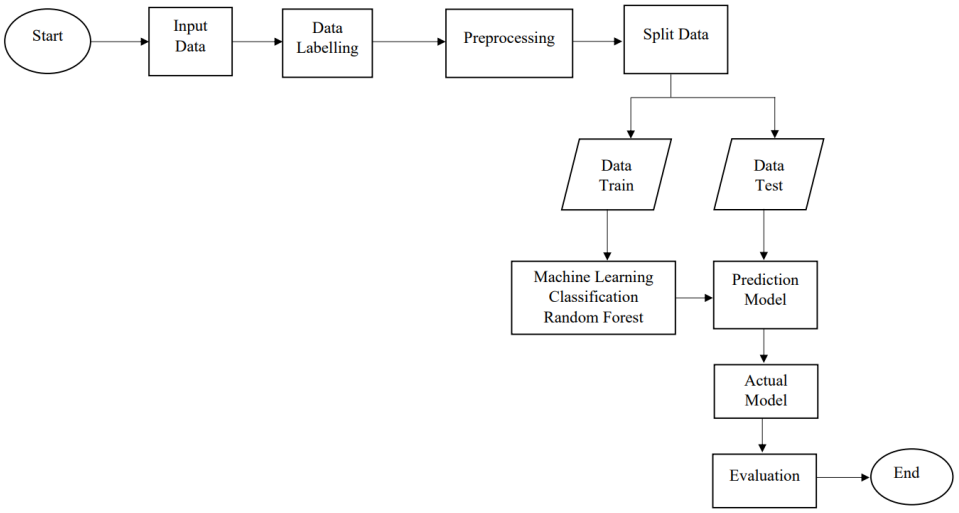
3.1. Pengumpulan Data

Dataset yang digunakan dalam penelitian implementasi multiprocessing pada klasifikasi data anemia adalah *Anemia Dataset* yang tersedia secara publik di Kaggle oleh BISWA RANJAN RAO. *Anemia Dataset* disajikan pada Tabel 1.

Tabel 1. Anemia Dataset

Gender	Hemoglobin	MCH	MCHC	MCV	Result
1	14.9	22.7	29.1	83.7	0
0	15.9	25.4	28.3	72	0
0	9	21.5	29.6	71.2	1
….	….	….	….	….	….
….	….	….	….	….	….
….	….	….	….	….	….
0	14.3	16.2	29.5	95.2	0
0	11.8	21.2	28.4	98.1	1

3.2. *Gambaran Sistem*



Gambar 1. Alur Klasifikasi

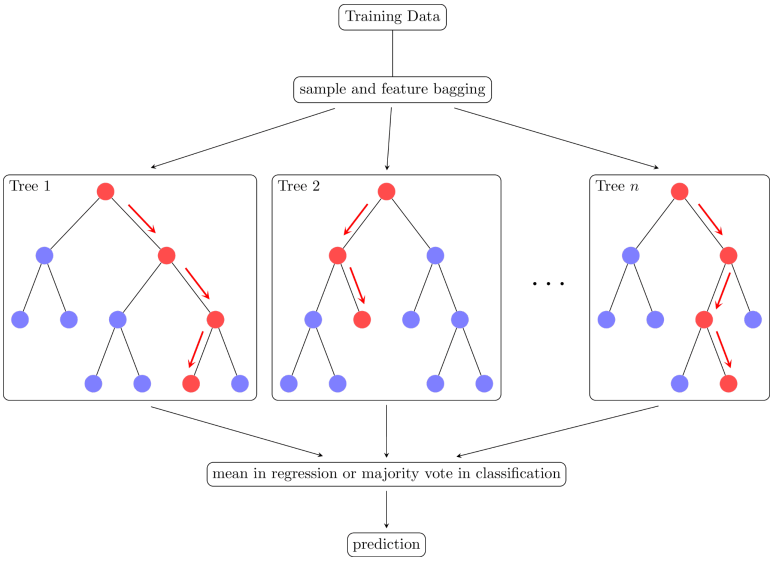
3.3. *Klasifikasi*

Proses *klasifikasi* merupakan suatu proses di mana suatu model pembelajaran mesin (*machine learning*) yang digunakan untuk memprediksi kelas atau label target berdasarkan fitur-fitur yang ada pada suatu data. Tujuan dari proses klasifikasi adalah untuk mengklasifikasikan atau mengelompokkan data ke dalam kategori atau kelas yang telah ditentukan sebelumnya. Dan terdapat beberapa algoritma yang dapat digunakan untuk proses *klasifikasi* diantaranya, yaitu *Naive Bayes, Decision Tree, Random Forest, Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Logistic Regression, Gradient Boosting, Neural Networks, AdaBoost,* dan *Gradient Descent*.

Pada penelitian kali ini, algoritma *Random Forest* yang digunakan untuk *klasifikasi.* *Random Forest* merupakan sebuah metode ensambel yang menggabungkan beberapa pohon keputusan (*decision tree*) secara paralel. Setiap pohon dalam *Random Forest* diperoleh dengan melakukan pembagian acak pada dataset yang berbeda dan menggunakan subset acak dari fitur yang tersedia. Pohon-pohon ini kemudian menghasilkan prediksi individu yang digabungkan untuk menghasilkan prediksi final.

Diagram algoritma hutan acak (*Random Forest*) (Breiman 2001). *Random Forest* adalah model ansambel yang terdiri dari pohon keputusan biner yang memprediksi mode prediksi pohon individu dalam klasifikasi atau rata-rata dalam regresi. Setiap node dalam pohon keputusan adalah kondisi pada fitur tunggal, yang dipilih untuk membagi kumpulan data menjadi dua sehingga sampel serupa berakhir di kumpulan yang sama. RF dapat diperiksa, invarian terhadap penskalaan dan transformasi fitur lainnya, kuat terhadap penyertaan fitur yang tidak relevan, dan dapat

memperkirakan kepentingan fitur melalui penurunan rata-rata ketidakmurnian (MDI).



Gambar 2. Diagram Random Forest

Proses klasifikasi terhadap algoritma *Random Forest* memiliki beberapa tahapan tersendiri, di antaranya yaitu :

a) Persiapan data
   Pada tahap ini merupakan tahapan persiapan data dengan melakukan penyesuaian format data agar sesuai dengan kebutuhan algoritmanya.
b) Pembuatan algoritma model Random Forest
   Pada tahap ini peneliti menyiapkan model RandomForestClassifier yang digunakan dalam ensemble yang dapat disesuaikan dengan kebutuhannya.
c) Pembagian Data
   Pada tahap ini dataset dibagi menjadi data latih (*training set*) dan data uji (*test set*).
d) Pelatihan model
   Pada tahap ini merupakan proses pembuatan decision trees yang menghasilkan Random Forest terdiri dari sekumpulan decision trees.
e) Prediksi
   Pada tahap ini dilakukan proses prediksi pada data tersebut dan hasilnya digunakan untuk analisis lebih lanjut pada proses evaluasi performa model.
f) Evaluasi model
   Pada tahap ini merupakan proses evaluasi dari model Random Forest menggunakan set pengujian.

g) Pengukuran performa
Pada tahap ini dilakukan pengukuran performa menggunakan matrik evaluasi yang relevan.

h) Output hasil
Output dari tahapan ini dapat digunakan untuk menganalisis dan mengevaluasi hasil klasifikasi.

4. Hasil dan Pembahasan

Dalam melakukan penelitian ini, penulis menggunakan bahasa pemrograman Python, Google Collab, dan beberapa library untuk menjalankan algoritma Random Forest.

```
df = pd.read_csv("anemia.csv")
df.head()
```

	Gender	Hemoglobin	MCH	MCHC	MCV	Result
0	1	14.9	22.7	29.1	83.7	0
1	0	15.9	25.4	28.3	72.0	0
2	0	9.0	21.5	29.6	71.2	1
3	0	14.9	16.0	31.4	87.5	0
4	1	14.7	22.0	28.2	99.5	0

Variabel df dibuat untuk menampung pembacaan file csv dan ditampilkan hanya 5 data teratas dengan menggunakan head().

```
df['Result'].replace(float('0'), 'no', inplace=True)
df['Result'].replace(float('1'), 'yes', inplace=True)
df['Gender'].replace(float('0'), 'male', inplace=True)
df['Gender'].replace(float('1'), 'female', inplace=True)
```

```
df.loc[(df['Gender'] == 'male') & (df['Hemoglobin'] <= 13.2), "status"] = "anemia"
df.loc[(df['Gender'] == 'female') & (df['Hemoglobin'] <= 11.6), "status"] = "anemia"
df.loc[(df['Gender'] == 'male') & (df['Hemoglobin'] >= 13.2), "status"] = "normal"
df.loc[(df['Gender'] == 'female') & (df['Hemoglobin'] >= 11.6), "status"] = "normal"
df.head()
```

	Gender	Hemoglobin	MCH	MCHC	MCV	Result	status
0	female	14.9	22.7	29.1	83.7	no	normal
1	male	15.9	25.4	28.3	72.0	no	normal
2	male	9.0	21.5	29.6	71.2	yes	anemia
3	male	14.9	16.0	31.4	87.5	no	normal
4	female	14.7	22.0	28.2	99.5	no	normal

Pertama, pada dataframe atribut result dan gender akan di replace menjadi yes dan no, serta male dan female, dan jika 0 menjadi no, 1 menjadi

yes. Selain itu, jika pria dewasa (male) dari 0 dan wanita dewasa (female) dari 1 . df.loc akan mengakses nilai-nilai dalam DataFrame yang akan menghasilkan hasil (Result dan Status) dari kadar hemoglobin yang <= 13.2 akan menghasilkan anemia pada pria dewasa, dan <= 11.6 akan menghasilkan anemia pada wanita dewasa. Begitu juga sebaliknya, jika >= 13.2 akan menghasilkan status normal pada pria dewasa, dan >= 11.6 akan menghasilkan status normal pada wanita dewasa.

```
df.to_csv('anemiadata.csv', index=False)
```

Proses dilakukan dengan menyimpan ‘df” ke dalam file CSV yang berjudul anemiadata.csv. Penulisan indeks di dalam file anemiadata.csv akan diabaikan karena diatur pada ‘index=False’.

```
# See the min, max, mean values
print('The highest hemoglobin was of:',df['Hemoglobin'].max())
print('The lowest hemoglobin was of:',df['Hemoglobin'].min())
print('The average hemoglobin in the data:',df['Hemoglobin'].mean())
```

```
The highest hemoglobin was of: 16.9
The lowest hemoglobin was of: 6.6
The average hemoglobin in the data: 13.412737508796623
```

Selanjutnya, untuk menentukan nilai max, min, dan rata-rata (mean) digunakan method max(), min(), dan mean(). Dari dataset anemia menghasilkan 16.9 untuk nilai max, 6.6 untuk nilai min, dan 13.412737508796623 untuk nilai mean.

```
from sklearn.ensemble import RandomForestClassifier
import multiprocessing
import time
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings("ignore")
```

```
data = pd.read_csv("anemiadata.csv")
data.head()
```

	Gender	Hemoglobin	MCH	MCHC	MCV	Result	status
0	female	14.9	22.7	29.1	83.7	no	normal
1	male	15.9	25.4	28.3	72.0	no	normal
2	male	9.0	21.5	29.6	71.2	yes	anemia
3	male	14.9	16.0	31.4	87.5	no	normal
4	female	11.1	23.3	29.5	79.1	no	normal

Kode di atas adalah bagian dari impor library dan pengaturan awal sebelum menjalankan algoritma RandomForestClassifier. Pafa ‘from

sklearn.ensemble import RandomForestClassifier' menjadi baris yang mengimpor kelas RandomForestClassifier dari modul ensemble di library scikit-learn. RandomForestClassifier adalah sebuah algoritma yang digunakan untuk melakukan klasifikasi menggunakan metode ensambel pohon keputusan. Pada 'import multiprocessing' menjadi baris yang mengimpor modul multiprocessing yang digunakan untuk mendukung pemrosesan paralel atau multiprocessing dalam Python. Pada 'import time' menjadi baris yang mengimpor modul time yang digunakan untuk mengukur waktu eksekusi program. Pada 'import pandas as pd' menjadi baris yang mengimpor modul pandas dan mengaliasnya sebagai pd. Pandas adalah library yang digunakan untuk manipulasi dan analisis data.

Pada 'import numpy as np' menjadi baris yang mengimpor modul numpy dan mengaliasnya sebagai np. Numpy adalah library yang digunakan untuk operasi numerik efisien dalam Python. Pada 'from sklearn.model_selection import train_test_split' menjadi baris yang mengimpor fungsi train_test_split dari modul model_selection di scikit-learn. Fungsi ini digunakan untuk membagi data menjadi data training dan data testing. Pada 'import warnings' menjadi baris yang mengimpor modul warnings yang digunakan untuk mengendalikan peringatan atau warning yang ditampilkan dalam program. Pada 'warnings.filterwarnings("ignore")' menjadi baris yang mengatur pengabaian peringatan yang akan ditampilkan dalam program. Dalam hal ini, semua peringatan akan diabaikan dan tidak ditampilkan.

Kode di atas mengimport library yang diperlukan untuk menggunakan RandomForestClassifier dan melakukan beberapa pengaturan awal seperti mengabaikan peringatan. Pengaturan ini membantu menjaga kelancaran eksekusi program dan memastikan bahwa peringatan yang mungkin muncul tidak mengganggu hasil akhir dari algoritma klasifikasi.

```
data['Hemoglobin'] = data['Hemoglobin'].astype(str)
data['Gender'] = data['Gender'].astype(str)
data['Result'] = data['Result'].astype(str)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1421 entries, 0 to 1420
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Gender      1421 non-null   object
 1   Hemoglobin  1421 non-null   object
 2   MCH         1421 non-null   float64
 3   MCHC        1421 non-null   float64
 4   MCV         1421 non-null   float64
 5   Result      1421 non-null   object
 6   status      1421 non-null   object
dtypes: float64(3), object(4)
memory usage: 77.8+ KB
```

Dalam DataFrame "data" pada kolom 'Hemoglobin', 'Gender', dan 'Result' akan diubah tipe datanya pada kolom tersebut menjadi tipe data

string. Kemudian, dipanggil dengan fungsi .info() yang akan menghasilkan beberapa output berupa informasi penting tentang dataset yang berisi jumlah total baris atau kolom pada DataFrame, nama dan tipe kolom, jumlah nilai non-null dalam tiap kolom, dan ringkasan penggunaan memori dari DataFrame.

```
data['status'].value_counts()
```

```
normal    957
anemia    464
Name: status, dtype: int64
```

Metode value_counts() digunakan untuk menghitung jumlah kemunculan pada kolom ‘status’ dalam objek DataFrame ‘data’.

```
results = []

def processData(data, outcome_filter):
    print(outcome_filter)
    df_filter=data[data['status']==outcome_filter]
    df_filter.drop(['status'], axis=1, inplace=True)
    data_new=df_filter
    clf=RandomForestClassifier(n_estimators=100)
    X=data_new.drop('Result', axis=1)  # Features
    y=data_new['Result']  # Labels

    # Split dataset into training set and test set
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1)
    clf.fit(X_train,y_train)
    y_out = clf.predict(X_test)
    X_test['status']=outcome_filter
    cols=list(X_test.columns) + ['Actual']
    out=pd.concat([X_test, y_test], axis=1, ignore_index=True)
    out.columns=cols
    out.reset_index(drop=True, inplace=True)
    final_out=pd.concat([out, pd.DataFrame(y_out)], axis=1, ignore_index=True)
    final_out.columns=list(out.columns)+['Prediction']
    print(final_out.shape)
    return final_out

def collect_results(result):
    results.extend(result.values.tolist())
```

```
multiprocessing.cpu_count()
```

```
2
```

Gambar 3. Proses Klasifikasi

Kemudian, terdapat variabel result dengan list kosong. Lalu, terdapat fungsi processData yang memiliki parameter data dana outcome_filter. Variabel df_filter digunakan untuk menampung data dengan atribut status yang akan dijadikan outcome_filter. df_filter akan menghapus kolom “status” dengan metode “drop” dengan menentukan axis=1 yang digunakan sebagai parameter. Hasil dari df_filter akan ditampung pada variabel data_new.

Variabel clf digunakan untuk menampung hasil dari RandomForestClassifier dengan parameter (n_estimators=100) yang memiliki arti 100 pohon keputusan akan dibentuk dan digabungkan untuk melakukan klasifikasi data. Dengan variabel x yang akan menghapus kolom “Result” dengan parameter axis=1 pada data_new yang akan menghasilkan DataFrame baru yang kolom tidak termasuk “result” dan variabel y kolom “result” akan diisi dengan data yang ada pada kolom “result” dari variabel data_new.

Algoritma RandomForestClassifier digunakan untuk melakukan proses klasifikasi yang berguna untuk memprediksi label ‘Result’ dari fitur-fitur yang terdapat dalam data. Model klasifikasi pada data akan dilatih dengan menggunakan data training (X_train dan y_train) dengan metode fit. Kemudian, pada data testing diprediksi (X_test) dengan metode predict. (y_out) atau hasil prediksi digabungkan dengan data testing (X_test) dan label sebelumnya (y_test) yang berguna untuk membentuk DataFrame final_out yang berisi kolom-kolom data testing. Dengan kolom ‘Actual’ berisi label sebenarnya, dan kolom ‘Prediction’ berupa prediksi hasil klasifikasi. Jadi, kode yang telah dibuat melibatkan proses klasifikasi untuk memprediksi label "Result" berdasarkan fitur-fitur pada data yang ada.

Actual dan Prediction akan menjadi tabel baru yang memiliki kegunaan sebagai berikut:

1. Actual: Kolom Actual digunakan untuk menyimpan nilai label sebenarnya (y_test) dari data uji (X_test). Nilai ini akan digunakan untuk membandingkan dengan hasil prediksi (y_out) untuk mengukur performa model.
2. Prediction: Kolom Prediction digunakan untuk menyimpan hasil prediksi (y_out) dari model klasifikasi. Nilai-nilai ini menunjukkan kelas prediksi untuk setiap data pada data uji.

Dengan memiliki kolom Actual dan Prediction, kita dapat membandingkan kelas yang diprediksi oleh model dengan kelas sebenarnya untuk mengevaluasi seberapa baik model dapat mengklasifikasikan data dengan benar. Evaluasi ini dapat dilakukan dengan berbagai metrik performa klasifikasi seperti akurasi, presisi, recall, atau F1-score. Pada akhirnya, hasil dari kolom Actual dan Prediction akan dikumpulkan dan disimpan dalam variabel results melalui fungsi collect_results.

Pada multiprocessing.cpu_count menghasilkan jumlah CPU yang tersedia ketika sistem komputer yang sedang berjalan. Pada program di atas menghasilkan angka 2, yang memiliki arti CPU yang digunakan ketika memproses dataset tersebut menggunakan 2 CPU. Jumlah CPU digunakan untuk pemanfaatan paralelisme atau penggunaan multiprocessing yang berguna untuk meningkatkan efisiensi dan kecepatan pemrosesan tugas yang dapat dipecah menjadi beberapa sub tugas secara independen.

```
if __name__ == "__main__":
    start_time = time.time()

    pool = multiprocessing.Pool(processes=multiprocessing.cpu_count()-1)
    for i in data['status'].unique():
        pool.apply_async(processData, args=(data.drop(['Hemoglobin', 'Gender'], axis=1), i), callback=collect_results)
    pool.close()
    pool.join()

    # Converts list of lists to a data frame
    dataFrame = pd.DataFrame(results, columns=list(data.drop(['Hemoglobin', 'Gender', 'Result'], axis=1).columns) + ['Actual', 'Prediction'])
    print("Dimensions in final test data {}".format(dataFrame.shape))
    print("--- %s seconds ---" % (time.time() - start_time))
```

```
normal
(96, 6)
anemia
(47, 6)
Dimensions in final test data (143, 6)
--- 0.602175235748291 seconds ---
```

Gambar 4. Proses Multiprocessing

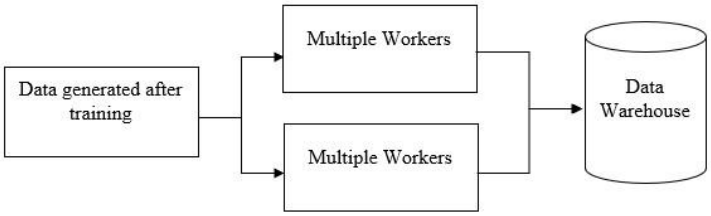
Kemudian dilakukan perhitungan waktu eksekusi program dengan memperhatikan file hanya dieksekusi pada blok atau dijalankan langsung. Pembuatan objek pool dari model multiprocessing yang digunakan untuk mengatur proses multi proses. Looping pertama dilakukan pengambilan nilai unik dari kolom ‘status’. Proses asinkron dilakukan oleh fungsi ‘processData’ dari setiap nilai ‘status’ yang dibaca sebagai argumen. Pada pool.close() sebagai tanda tidak adanya proses yang nantinya akan ditambahkan ke dalam pool. Pada pool.join() sebagai tanda untuk menunggu semua proses selesai dieksekusi yang selanjutkan akan dieksekusi ke dalam baris selanjutnya. Akhir dari proses klasifikasi, outputnya akan dikumpulkan dari nilai ‘status’ menggunakan fungsi ‘collect_results’. Variabel ‘results’ akan menyimpan sekumpulan output. Variabel dataFrame akan menyimpan kolom dari data kecuali kolom ‘Hemoglobin’, ‘Gender’, dan ‘Result’ dan ditambahkan data berupa kolom ‘Actual’ dan ‘Prediction’. Selanjutnya ditampilkan hasil dimensi berupa jumlah baris dan kolomnya, serta ditampilkan total waktu eksekusi program dalam satuan detik.

```
dataFrame
```

	MCH	MCHC	MCV	status	Actual	Prediction
0	21.5	30.4	92.7	normal	no	no
1	19.5	28.7	87.7	normal	yes	yes
2	19.4	29.4	88.2	normal	no	no
3	18.7	30.3	69.4	normal	no	no
4	27.6	29.3	77.8	normal	no	no
...	...	...	...	...	...	...
138	22.5	32.1	75.0	anemia	yes	yes
139	18.5	29.3	93.9	anemia	yes	yes
140	28.6	28.6	84.8	anemia	yes	yes
141	27.5	28.2	93.0	anemia	yes	yes
142	25.0	30.4	81.4	anemia	yes	yes

143 rows × 6 columns

Proses terakhir merupakan menampilkan hasil dari variabel dataFrame yang berisi hasil klasifikasi dengan informasi data dari kolom MCH, MCHC, MCV, status, actual, dan prediction.



Dengan menggunakan modul multiprocessing, kita dapat menjalankan fungsi secara paralel menggunakan beberapa proses, memanfaatkan potensi pemrosesan paralel pada sistem dengan CPU yang tersedia. Bagan di atas menjelaskan bahwa pekerjaan akan dibagi ke beberapa worker yang tersedia dan kasus di atas CPU yang tersedia adalah 2 maka worker dibagi menjadi 2 bagian.

5. Kesimpulan

Data yang telah diolah di atas merupakan dataset anemia yang diklasifikasikan dengan menggunakan metode multiprocessing dengan algoritma Random Forest. Preprocessing, dengan menggunakan pandas data dapat dibaca dan dilakukan beberapa operasi preprocessing dengan mengganti nilai-nilai tertentu dengan kategori yang lebih jelas (0 menjadi ‘no’ dan 1 menjadi ‘yes’). Pembentukan model merupakan hasil dari data yang sudah di preprocessing yang dibagi menjadi fitur dan label. Dalam melakukan model Random Forest Classifier dapat digunakan library scikit-learn RandomForestClassifier. Proses klasifikasi dilakukan dengan multiprocessing yang proses pengerjaannya dilakukan secara paralel dari subset data yang berbeda. Pada tiap subset data terdapat fungsi processdata yang akan melakukan pelatihan model dan prediksi dalam subset tersebut. Hasil eksekusi akan disimpan di variabel results menggunakan fungsi collect_results. Proses klarifikasi berakhir dengan dilakukannya konversi hasil prediksi dan label ke dalam dataFrame dan ditampilkan dalam bentuk jumlah baris, jumlah kolom, dan waktu eksekusi proses eksekusi program.

Program mengimplementasikan klasifikasi menggunakan konsep multiprocessing dengan algoritma Random Forest pada dataset anemia. Proses klasifikasi dilakukan secara paralel sehingga efisiensi dapat meningkat. Penerapan multiprocessing dapat meningkatkan kecepatan pemrosesan dan kemungkinan penanganan dataset dalam jumlah yang lebih besar.

6. Referensi

HENDRIAN, S. (2018). *Algoritma Klasifikasi Data Mining untuk Memprediksi Siswa dalam Memperoleh Bantuan Dana Pendidikan*.

Ramli, R. G., & Sibaroni, Y. (2022, Februari 1). Klasifikasi Topik Twitter menggunakan Metode Random Forest dan Fitur Ekspansi Word2Vec. *e-Proceeding of Engineering*, *9*, 79.

Raza, Z., Haq, I. u., Muneeb, M., & shafiq, O. (2021, Oktober 31). Energy Efficient Multiprocessing Solo Mining Algorithms for Public Blockchain Systems. *Scientific Programming*, 13.

RIYANTO, U. (2018, Oktober). Analisis Perbandingan Algoritma Naive Bayes dan Support Vector Machine dalam Mengklasifikasikan Jumlah Pembaca Artikel Onlie. *Jurnal Teknik Informatika (JIKA) Universitas Muhammadiyah Tangerang*, 63.

Wibowo, I. K., Anom Besari, A. R., & Rizqullah, M. R. (2021, Januari 1). Implementation of Multiprocessing and Multithreading for End Node Middleware Control on Internet of Things Devices. *Jurnal Ilmiah Bidang Teknologi Informasi dan Komunikasi*, *6*, 54-60.

Yovellia Londo, G. L., Purnomo W.P, Y. S., & Maslim, M. (2020, Mei). Pembangunan Aplikasi Identifikasi Kesalahan Ketik Dokumen Berbahasa Indonesia Menggunakan Algoritma Jaro-Winkler Distance. *Jurnal Informatika*, *8*, 19-27.