Prediksi Mahasiswa Dropout Atau tidak menggunakan Long Short Term Memory(LTSM) dan Multi-Layer Perceptron(MLP)

Sebuah Makalah
Ditujukan sebagai
tugas project case based pembelajaran mesin

oleh:

Nama	***Nim***
1. *Quinta Bilqis Kharisma*	*1304211026*
2. *Akmal Taufik Fadhlurohman*	*1304211028*
3. *Muhammad Affiq Fikri*	*1304211029*
4. *Khalifardy Miqdarsah*	*1304211035*
5. *Firdha Agustya Rimawan*	*1304211042*

Dosen:
SITI SA'ADAH (SSD)

PROGRAM STUDI S1 PJJ TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM
BANDUNG

2023

Lembar Pernyatan

Pernyataan:

Tim Kami mengerjakan tugas ini dengan cara yang tidak melanggar aturan perkuliahan dan kode etik akademisi. Jika melakukan plagiarism atau jenis pelanggaran lainnya, maka Tim kami bersedia diberi nilai E untuk Mata Kuliah ini.

Kata Pengantar

Syukur yang paling utama kami panjatkan kepada Allah Swt, atas segala rahmat, cinta dan kasih sayang yang tak terbatas. Kepada kekasihnya yang utama, manusia terbaik, yang tanpanya alam semesta tak akan diciptakan , kanjeng nabi muhammad SAW. Semoga umatmu yang bodoh ini tak membuat wajahmu berpaling kehilangan senyuman yang kami nantikan. Tak lupa juga kepada orang tua yang mengandung dan merawat kami, hormat dan cinta kami untuk kalian. Seluruh guru-guru kehidupan, orang-orang terpinggirkan dan mereka yang terasingkan oleh zaman. Serta pada alam semesta itu sendiri yang tak henti-hentinya membuat pikiran takjub, yang rela menjadi wadah bagi seongok makhluk yang meskipun paling bungsu dan lemah namun sangat angkuh kepada akalnya sendiri, padahal mereka tidak ada apa-apanya dibandingkan dengan hamparan semesta.

Bekasi, 04 Nopember 2023

Tim Penulis

Daftar isi

Daftar Gambar

Daftar Tabel

Bab 1

Pendahuluan

Pada study case makalah ini dataset mengambil dari situs ***https://archive.ics.uci.edu/dataset/697/predict+students+dropout+and+academic+success.*** Datset tersebut berisi kumpulan data akademik dengan fitur-fiturnya seperti *marital status*, Application mode, Application order, Course, daytime/evening attedance, previous qualification dan sebagainya. Kumpulan fitur tersebut bisa dilihat pada gambar satu berserta dengan tipe datanya.

```
Data columns (total 37 columns):
 #   Column                                          Non-Null Count  Dtype
---  ------                                          --------------  -----
 0   Marital status                                  4424 non-null   int64
 1   Application mode                                4424 non-null   int64
 2   Application order                               4424 non-null   int64
 3   Course                                          4424 non-null   int64
 4   Daytime/evening attendance                      4424 non-null   int64
 5   Previous qualification                          4424 non-null   int64
 6   Previous qualification (grade)                  4424 non-null   float64
 7   Nacionality                                     4424 non-null   int64
 8   Mother's qualification                          4424 non-null   int64
 9   Father's qualification                          4424 non-null   int64
 10  Mother's occupation                             4424 non-null   int64
 11  Father's occupation                             4424 non-null   int64
 12  Admission grade                                 4424 non-null   float64
 13  Displaced                                       4424 non-null   int64
 14  Educational special needs                       4424 non-null   int64
 15  Debtor                                          4424 non-null   int64
 16  Tuition fees up to date                         4424 non-null   int64
 17  Gender                                          4424 non-null   int64
 18  Scholarship holder                              4424 non-null   int64
 19  Age at enrollment                               4424 non-null   int64
 20  International                                   4424 non-null   int64
 21  Curricular units 1st sem (credited)             4424 non-null   int64
 22  Curricular units 1st sem (enrolled)             4424 non-null   int64
 23  Curricular units 1st sem (evaluations)          4424 non-null   int64
 24  Curricular units 1st sem (approved)             4424 non-null   int64
 25  Curricular units 1st sem (grade)                4424 non-null   float64
 26  Curricular units 1st sem (without evaluations)  4424 non-null   int64
 27  Curricular units 2nd sem (credited)             4424 non-null   int64
 28  Curricular units 2nd sem (enrolled)             4424 non-null   int64
 29  Curricular units 2nd sem (evaluations)          4424 non-null   int64
 30  Curricular units 2nd sem (approved)             4424 non-null   int64
 31  Curricular units 2nd sem (grade)                4424 non-null   float64
 32  Curricular units 2nd sem (without evaluations)  4424 non-null   int64
 33  Unemployment rate                               4424 non-null   float64
 34  Inflation rate                                  4424 non-null   float64
 35  GDP                                             4424 non-null   float64
 36  Target                                          4424 non-null   object
```

Gambar 1.1. Tabel Fitur

Dari dataset tersebut akan dibuat suatu model machine learning yang bisa memprediksi apakah seorang mahasiswa akan berpotensi dropout kedepannya atau sukses secara akademik. Sehingga diharapkan angka putus sekolah ataupun kegagalan akademik di pendidikan tinggi bisa berkurang.

Secara kualitas dataset sudah cukup baik karena pada dasarnya data set yang diambil pada situs sudah di lakukan cleaning sehingga sudah tidak terdapat data missing ataupun outlier yang cukup banyak. Meskipun begitu eksplorasi data tetap dilakukan untuk mencari titik-titik atau bagian-bagian yang bisa meningkatkan kualitas dari data.

Pada mulanya di lihat terlebih dahulu bagaimana perbandingan jumlah data masing-masing terhadap jumlah class di dalam kolom target di dapatkan sebagai berikut:

```
Target

Graduate 2209

Dropout 1421

Enrolled 794

Name: count, dtype: int64
```

Jika dibandingkan data enrolled jauh lebih sedikit dibanding data lainnya sehingga perlu dilakukan balancing supaya jumlah data seimbang.

Data perlu di balancing karena data yang imbalance akan mempengaruhi performansi model yang nantinya akan dibangun.

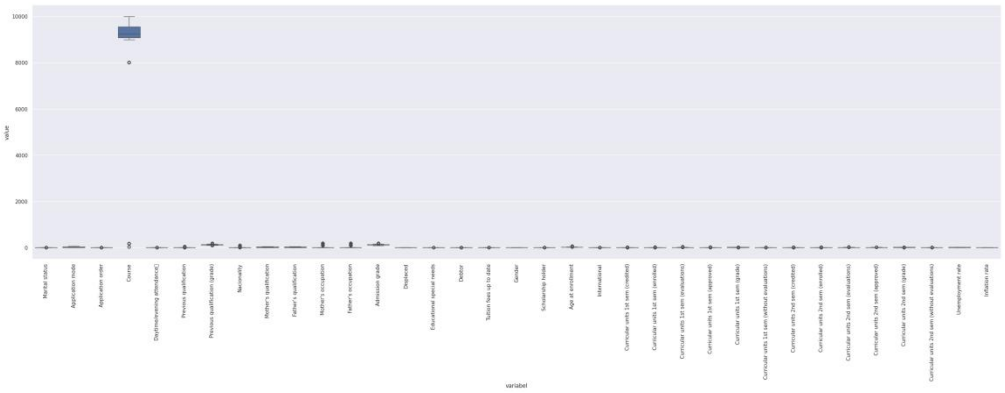
Kemudian dilakukan observasi bagaimana distribusi pada setiap fitur di dataset hal ini akan mempengaruhi ketika melakukan standarisasi apakah akan menggunakan standarisasi dengan menggunakan mean dan standard deviasi atau minmax. Hasil dari penggambaran distribusi bisa dilihat pada gambar 2.

Gambar 1.2. Distribusi masing-masing fitur

Bisa dilihat pada gambar 2 sebagian besar fitur terdistribusi normal, meskipun memang terdapat beberapa fitur terutama pada data-data yang bersifat boolean ataupun kategorical, seperti marital status, gender, nasionality dsb berdistribusi tidak

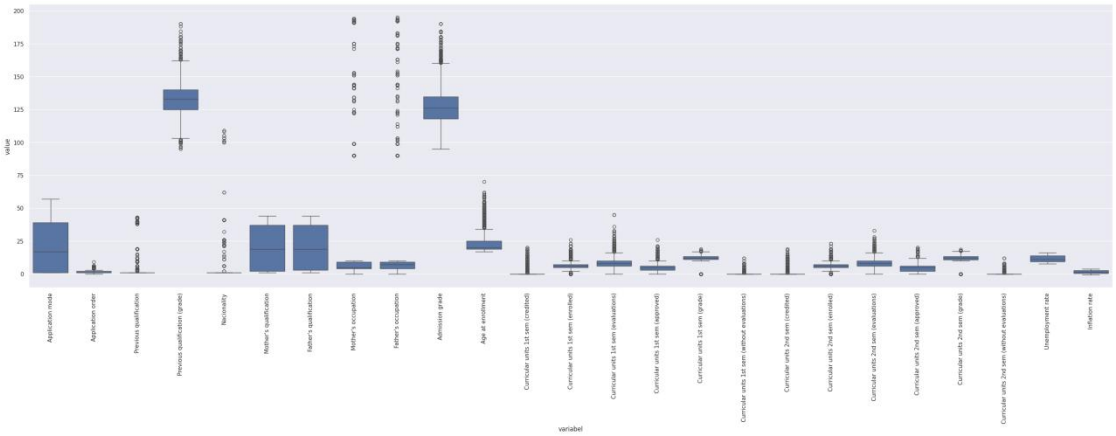
normal. Namun pada umunya tipe data seperti ini akan di drop out karena tidak berpengaruh signifikan terhadap kumpulan data.

Setelah itu dilakukan observasi terhadap outliers dengan menggunakan box plot sehingga akan dikethui seberapa banyak outliers pada masing-masing fitur. Box plot bisa dilihat pada gambar 3.



Gambar 1.3. Boxplot masing-masing fitur

Jika dilihat pada boxplot di gambar 3 seakan-akan semua fitur tidak memiliki outlier namun hal ini dikarenakan skala yang cukup besar dari course sehingga fitur lainnya yang mempunyai skala yang jauh lebih kecil menjadi tidak terlalu terlihat, untuk memperlihatkan boxplot pada fitur lainnya, kami mengabaikan fitur course untuk membuat boxplo, hasilnya bisa dilhat pada gambar 4.



Gambar 1.4. Boxplot tanpa course

Boxplot pada gambar 4 ini terlihat jelas bahwa pada beberpa fitur tetap memiliki cukup banyak outliers bahkan hampir sebagian besar memiliki outliers. Outliers ini pada akhirnya nanti menentukan bagaimana standarisasi akan dilakukan.

Selanjutnya yang dilakukan adalah melihat keterhubungan atau korelasi antara setiap fitur dengan target labelnya. Dikarenakan kolom yang akan diprediksi mempunyai tipe data yang kategorikal dan untuk variabel indepeden mempunyai data numerik dan boolena. Maka uji korelasi menggunakan Metode Annova dan point biserial untuk boolean.

Pada dasarnya kedua metode tersebut adalah menghitung p-value, lalu berdasarkan p-value tersebut dan parameter yang ditentukan akan disimpulkan seberapa besar korelasi antar variabel dan kolom target. Pada makalah ini nilai p-value bisa dilihat pada gambar 5.

Gambar 1.5. p-value fitur dengan kolom target

Umumnya nilai batas p-value adalah yang lebih kecil sama dengan dari 0.05 (P-value $\leq$ 0.5) yang memiliki nilai koreasi yang kuat, sehingga fitur fitur yang nilai p-valuenya lebih dari 0.5 (p-value > 0.5) artinya tidak memiliki korelasi dengan kolom target sehingga pada tahap pre processing bisa diabaikan.

Bab 2

Pre-processing

Setelah di lakukan eksplorasi data pada tahap sebelumnya untuk melihat bagaimana karakteristik pada suatu kumpulan data, pada tahap ini berdasarkan eksplorasi sebelumnya akan dilakukan pre-processing data untuk membuat kualitas data set lebih baik.

Pada eksplorasi data kita mengetahui untuk beberapa fitur yang memiliki p-value lebih kecil sama dengan dari 0.05 bisa di drop beberapa fitur tersebut adalah :

1. Educational special needs
2. Inflation rate
3. GDP
4. Unemployement rate
5. Previous qualification
6. Course
7. International
8. Nacionality

code yang diimplementasikan adalah sebagai berikut:

```
kolom_drop = ["Educational special needs", "International","Nacionality",'Inflation rate','Unemployment rate','GDP','Previous qualification','Course']

data_fitur = data_fitur.drop(columns=kolom_drop)
```

kolom_drop adalah suatu variabel yang menyimpan data list berisi kumpulan nama fitur yang akan di drop dari dataset.

data_fitur adalah suatu variabel yang sebelumnya menyimpan dataset dari **data.csv** lalu dikonversi kedalam bentuk dataframe pandas. method **.drop** adalah method pada dataframe pandas untuk menghapus fitur dari dalam dataset **method** tersebut memiliki parameter berupa **columns**

Selain itu di karenakan terdapat imbalance data pada dataset maka dilakukan balancing data dengan cara undersampling, artinya memotong data yang mayoritas sehingga sama dengan minoritas. Metode ini dilakukan meskipun mungkin ada beberapa karakteristik data yang mungkin ikut hilang bersama dengan data yang dihapus namun karena jumlah sampel minoritas cukup banyak walaupun sedikit dibandingkan dengan jumlah data kelas mayor, harapanya tidak terlalu berpengaruh terhadap performansi model adapun implementasi codenya sebagai berikut:

```
from imblearn.under_sampling import RandomUnderSampler

under = RandomUnderSampler(sampling_strategy='majority')
```

```
data_fitur,target = under.fit_resample(data_fitur,target)
```

RandomUnderSampler adalah salah satu class untuk undersampling dari library imblearn. Pada code diatas class ini dipanggi dengan parameter **sampling_startegy** = ‘majority’ , yang artinya data mayoritas akan dipotong sesuai dengan jumlah data minoritas. lalu hasil dari samplingnya akan mengahsilkan tupple dengan data_fitur tanpa target dan target.

Selanjutnya pada eksplorasi sebelumnya ditemukan juga terdapat beberapa fitur memiliki outliers sehingga akan dilakukan perhitungan nilai zscores untuk menentukan outliers mana yang akan di drop dari dataset. zscores sendiri adalah suatu ukuran pada data seberapa dekat suatu data pada rata-ratanya. Nilai zscores bisa negatif atau positif , yang jika secara absolut , artinya nilai negatif akan dijadikan positif , semakin besar nilai zscores maka semakin jauh dari nilai rata-rata. Data yang memiliki nilai zscores yang besar ini lah yang disebut dengan outlier.

Dengan nilai zscore inilah akan di drop data-data dengan nilai absolut zscore kurang dari 5. Penentuan nilai 5 disini sebenarnya tidak ada standard yang pasti tapi kami sengaja memilih 5 supaya data outlier benar-benar terhapus dan tidak menghapus data yang bukan outliers. Untuk implementasi codenya adalah sebagai berikut:

```
#kick outlier zscores <5


z = np.abs(stats.zscore(data_fitur[data_fitur.columns]))

data_fitur = data_fitur[(z<5).all(axis=1)]
```

z adalah suatu variabel yang menyimpan dataframe yang berisi nilai zscore pada setiap record pada data_fitur.

stats.zscore() adalah fungsi library dari **scipy** untuk menghitung zscores dengan inputan berupa dataset yang ingin diukur.

pada baris kedua mengimplmentasikan bahwa data_fitur akan menyimpan setiap record yang memiliki nilai z-score dibawah 5.

Langkah selanjutnya adalah melakukan standarisasi sehingga skala nilai pada setiap fitur sama. Karena skala yang berbeda-beda akan menyebabkan penurunan performansi pada model machine learning. Metode yang dipakai untuk standarisasi kali ini adalah menggunakan metode standarisasi dengan mean dan standard deviasi. Adapun formula matematis nya adalah:

$$X' = \frac{X - \mu}{\sigma}$$

dimana :

X’ = data hasil standarisasi

X = data awal

μ = rata-rata

σ = standard deviasi

Pemilihan metode standarisasi dengan menggunakan nilai rata-rata dan standard deviasi disebakan hampir sebagian besar fitur memiliki distribusi normal.

Untuk implementasi codenya adalah sebagai berikut:

```
scaler = StandardScaler()

data_scaled = scaler.fit_transform(data_fitur)
```

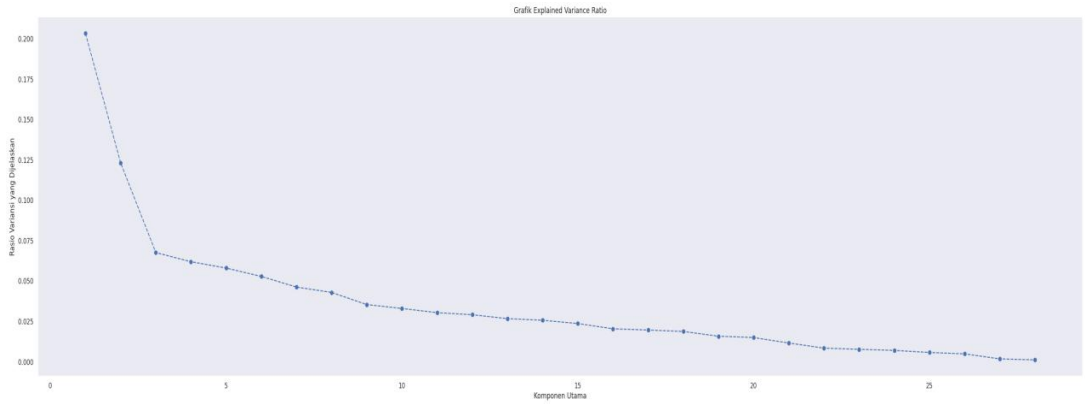
scaler menyimpan suatu object **StandardScaler()** suatu kelas di dalam library scikit-learn yang mempunyai fungsi untuk standarisasi dengan mean dan standard deviasi.

data_scaled variabel yang menyimpan hasil **standarisasi**

Setelah standarisasi selanjutnya adalah mereduksi dimensi dengan menggunakan PCA (principal component analisis). PCA adalah suatu metode untuk memilih fitur-fitur atau komponen-komponen yang paling utama dari fitur. Pengunaan PCA ini dilakukan supaya mengurangi data-data redudan dan mempercepat komputasi karena tidak semua fitur dipakai untuk membuat model.

Pertama-tama yang harus dilakukan adalah memilih berapa komponen yang akan dipakai atau dipilih. Untuk menentukan berapa komponen yang akan dipilih salah satu caranya dalah melihat variansi rasio. Variansi rasio adalah suatu nilai yang menunjukan seberapa banyaknya pemilihan komponen berpengaruh pada variansi. Untuk menentukan seberapa banyak yang akan dipilih maka tergantung kasus atau dari penggunaan seberapa banyak atau kepercayaan terhadap suatu data set akan dipilih. Jika kepercayaan atau data yang ingin digambarkan sebanya 95 %, maka jumlahkan setiap variasni rasio pada setiap titik banyaknya komponen hingga berjumlah 95 % . pada titik dimana rasio sudah 95 % pada titik tersebutlah berpa komponen yang akan diambil.

Untuk kasus pada makalah ini variansi rasio digambarkan pada grafik di gambar 6.



Gambar 2.1. Variansi Rasio

Jika dilihat pada grafik karena kasus pada makalah ini juga memakai kepercayaan sebanyak 95 %, maka banyaknya komponen yang diambil adalah sebanyak 20 komponen, sehingga implementasi code nya akan sebagai berikut.

```
pca2 = PCA(n_components=20)

pca2.fit(data_scaled)

transformed_data2 = pca2.transform(data_scaled)
```

pca2 adalah variabel yang menyimpan objek **PCA** dengan parameter **n_components** diisi 20 karena kompnen yang akan diambil sebanyak 20. **PCA** sendiri adalah class di library **scikit-learn**

transformed_data 2 adalah variabel yang menyimpan data hasil **PCA**

Setelah data sudah transform selanjutnya data hasil transform tersebut di rubah kedalam bentuk dataframe pandas lalu ditambahkan kolom target berserta valuenya. setelah itu simpan dataframe tersebut dalam bentuk file csv. Implementasi codenya adalah sebagai berikut.:

```
data_pca = pd.DataFrame(data=transformed_data2, columns=[f"PCA{i+1}" for i in
range(20)])

data_pca["Target"] = dataku["Target"]

data_pca.to_csv('data_eda.csv',index=False)
```

data_pca adalah suatu variabel yang menyimpan dataframe hasil PCA dimana columns nya adalah **PCA1 - PCA20.**

Pada baris kedua arti code tersebut adalah menambahkan kolom target pada dataframe **data_pca** dengan value pada kolom **target** di datafrane **dataku**.

Baris ketiga adalah code untuk menyimpan data frame dalam bentuk csv , index = False , supaya index pada datra frame tidak ikut disimpan.

Dengan file csv sudah disimpan maka proses pre-processing sudah selesai. file csv akhir inilah yang nantinya akan di train untuk dibuat model machine learning.

Bab 3

Implementasi Algoritma

Seperti yang telah disampaikan pada bab 1 bahwasannya studi kasus pada makalah ini adalah bertujuan untuk memprediksi dengan data fitur yang relevan apakah seorang akan dropout pada perkuliahan atau tidak. Oleh karena itu dirancanglah suatu solusi berbasis pembelajaran mesin yang sesuai pada kasus ini.

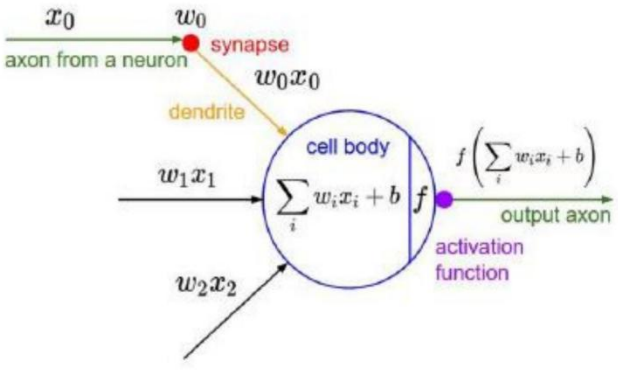
Pemilihan algoritma machine learning yang sesuai di landaskan pada tujuan dari masalah yang ingin dipecahkan dan karakteristik data. Karena tujuan dari masalah di makalah ini prediksi maka ada beberapa algoritma yang cocok untuk dipakai diantaranya KNN, Naive-bayes, Artificila neural network (ANN), Support Vektor machine (SVM) dan recurent neural network (RNN). Namun karena batasan masalah pada makalah ini dibatasi pada metode SVM, ANN dan RNN saja, maka yang akan dipakai adalah ANN dengan turunannya yaitu multi layer perceptron (MLP) dan RNN dengan turunannya Long-short Term memory (LSTM).

Pemilihan MLP sebagai hipotesis untuk memecahkan masalah prediksi mahasiswa dropout di dasari pada dataset yang cukup banyak, kompleks dan secara natural dataset tidak terpisah secara linear. Dibandingkan dengan SVM yang memang baik meklasifikasikan atau memprediksi data jika terpisah secara linier dan data set tidak terlau kompleks dan banyak. Sedangkan pemilhan LTSM dipilih sebagai model pembanding yang berbasis deep learning .

Pada bab ini akan dijelaskan bagaimana implementasi Algoritma MLP dan LTSM pada code python untuk menangani studi kasus prediksi mahasiswa dropout. Namun sebelum itu akan di jelaskan terlebih dahulu landasan teori bagaimana MLP dan LTSM bekerja.

3.1. Multi-Layer Perceptron (MLP)

MLP adalah pengembangan dari artificial neural network yang mengadopsi bagaimana sistem saraf pada manusia bekerja. Sebuah sel saraf utuh memiliki dendrite sebagai masukan, lalu cell body dan axon sebagai output. Begitu juga pada abstraksi dari artificial neural network. Ada input dengan bobot yang menyertainya lalu body cell dengan fungsi sigma dan output yang berupa fungsi aktivasi. Ilustrasi bisa dilihat pada gambar 7.



Gambar 3.1. ilustrasi artficial neural network

Yang dimaksud dengan perceptron adalah seperti yang diilustrasikan pada gambar 7. Sebuah perceptron tunggal berisi *linier combiner* disertai dengan *hard limiternya.* linier combiner adalah penjumlahan hasil dari perkalian bobot (w) dengan input (x) ditambahkan dengan biasnya. Formulasinya sebagai berikut:

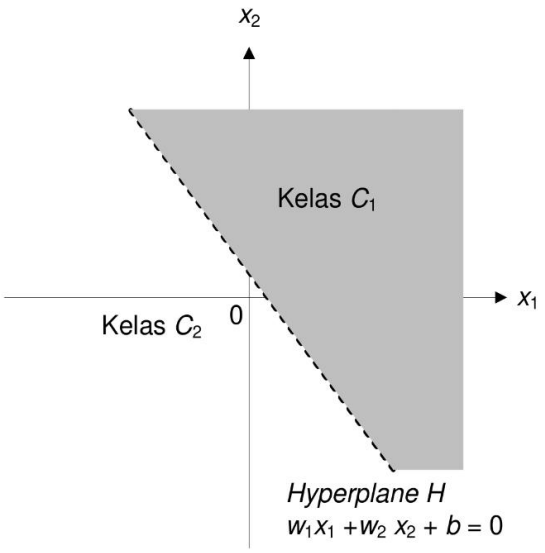
$$\text{sum\_function} = \sum_{i=1}^{n} w_i x_i + b$$

Dengan melewatkan hasil sum_function kedalam hard limiter berupa activation funtion suatu perceptron tunggal bisa mengklasifikasikan suatu inputan kedalam suatu kelas C1 atau C2 sesuai dengan aturannya.

Pada dasarnya perceptron adalah mencari suatu hyperplane / batas yang memisahkan data sesuai dengan kelasnya. Sehingga jika suatu data baru akan diklasifikasikan sesuai dengan sisi dimana hyperplane memisahkannya. Formula hyperplane yang memisahkan data tersebut dituliskan

$$\sum_{i=1}^{n} w_i x_i + b = 0$$

Contoh pada gambar 8 ditunjukan bagaimana dua variabel input bisa dibedakan pada suatu hyperplane berupa garis lurus.



Gambar 3.2. Ilustrasi hyperplane

Selanjutnya untuk setiap input x(n) dan $x_0 = 1$ dapat ditulis :

$$x(n) = [\ 1, x_1(n), x_2(n),....,x_m(n)]^T$$

Dan untuk setiap w(n) dan $w_0 = b$ dapat di tuliskan:

$$w(n) = [\ b, w_1(n), w_2(n),....,w_m(n)]^T$$

Sehingga sum function bisa di tuliskan:

$$v(n) = \sum_{i=0}^{m} w_i(n)x_i(n) = w^T(n) \cdot x(n)$$

Dimana

$w^T x > 0$

untuk setiap vektor input x berada di kelas C_1 dan

$w^T x \leq 0$

untuk setiap vektor input x berada di kelas C_2

Dari sini bisa dilihat bahwa hyperplane atau garis keputusan tergantung kepada w yang tepat sehingga hyperplane tepat memabagi data pada dua class.

ANN intinya adalah pembelajaran pada perceptron menghasilkan w yang tepat sehingga pertidaksamaan sebelumnya terpenuhi. Jika suatu input vektor x pada bobot w sudah diklasifikasikan dengan benar maka tidak ada koreksi tetapi jika kurang tepat maka w akan diupdate dengan persamaan :

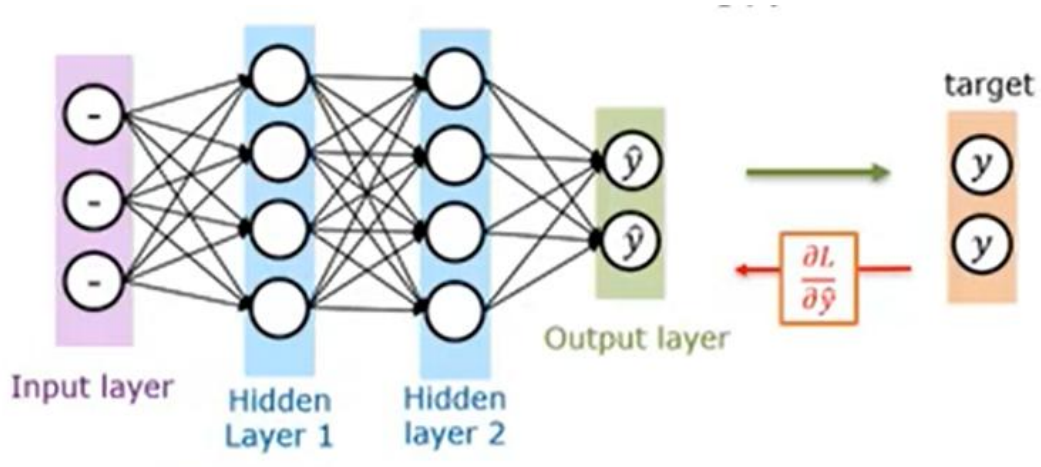
$$w(n+1) = w(n) - \eta(n)\, x(n) \text{ untuk } w^T x > 0$$

dan

$$w(n+1) = w(n) + \eta(n)\, x(n) \text{ untuk } w^T x \leq 0$$

$\eta(n)$ adalah suatu konstanta yang mewakili learning-rate , untuk merubah w pada setiap iterasi.

Lalu yang dimaksud dengan MLP adalah ANN yang memiliki banyak layer tidak hanya ada layer input lalu output namun terdapat yang dinamakan dengan hidden layer. Setiap hidden layer bisa memiliki lebih dari satu perceptron dan bisa memiliki lebih dari satu output.



Gambar 3.3. Ilustrasi multi layer perceptron

3.2. Long short term memory (LSTM)

LSTM adalah pengembangan dari RNN(recurrent neural networks, permasalahan pada RNN saat gradient semakin mengecil seiring dengan banyaknya layer, ditangani dengan baik dengan metode LSTM.

Untuk menjelaskan LSTM akan dijelaksan terlebih dahulu tentang RNN. RNN adalah metode dalam machine learning yang menangani data sekuensial. artinya data yang memiliki fitur-fitur yang saling terkait dengan waktu, contohnya adalah data ramalan cuaca dan data text.

RNN sama seperti artificial neural network pada umumnya, RNN memiliki perceptron dengan inputan dan outputan yang akan diaktivasi oleh fungsi aktivasi, yang membedakan adalah pada RNN terdapat suatu lopping atau feedback yang akan menjadi inputan selanjutnya . Looping tersebut berisi infromasi state terdahulu yang di formulasikan sebagai berikut:

$$h_t = f_w(h_{t-1}, x_t)$$

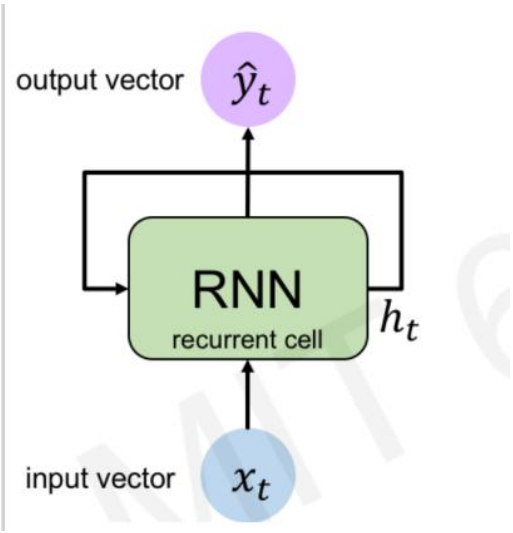
dimana:

h_t = state saat ini

f_w = fungsi bobot w

h_{t-1} = state sebelumnya

x_t = input vektor pada waktu ke t

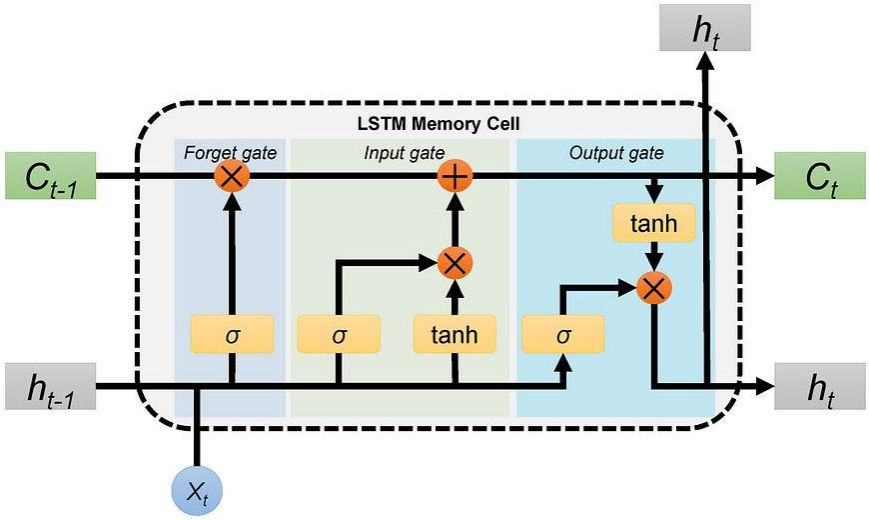


Gambar 3.4. Ilustrasi Cell RNN

Seperti yang sudah disebutkan sebelumnya masalah utama RNN adalah hilangnya gradient seiring dengan data sekuensial yang semakin panjang, untuk itu LSTM hadir untuk menangani masalah tersebut.

LSTM memiliki 3 gates yaitu input, forget dan output. Gerbang forget berfungsi untuk menentukan infromasi mana yang perlu diingat dan dilupakan, gerbang input bertugas untuk memilih informasi mana yang harus ditambahkan

kedalam cell LSTM. Sedangkan gerbang output menentukan output apa yang keluar dari cell berdasarkan nilai sel memori yang telah diperbarui.



Gambar 3.5. ilustrasi cell LSTM

3.3. Implementasi

Pengunaan MLP sebagai metode untuk memprediksi mahasiswa yang dropout diharapkan akan mempunyai akurasi 80-90 % karena memang MLP mempunyai kelebihan pada kemampuan untuk mengklasifikasikan dengan baik. Sedangkan penggunaan LSTM kemungkinan besar akurasinya akan dibawah dari MLP, karena dari data set yang ditangani bukan jenis dataset sequensial yang bisa di klasifikasikan dengan baik oleh model RNN atau LSTM sehingga besar kemungkinan akurasi LSTM akan berada di bawah MLP.

3.3.1. Implementasi MLP

Pada mulanya ditentukan terlebih dahulu hyperparameter yang akan digunkana pada implementasi MLP. hyperparameter sendiri tedapat dua jenis optimizer parameter dan model parameter. Adapun yang nanti akan dibandingkan hasilnya adalah optimizer parameter khususnya di learning rate, sedangkan parameter lainnya dibuat tetap. Adapun pemilihan parameter lebih rinci pada tabel berikut:

Parameter	Value
learning rate	0.001
batch	1
epochs	200
hidden unit	10
input first layer unit	sebanyak jumlah data set
Banyak nya layer	12

Table 3.3.1. Hyperparameter MLP

Learning rate pada mulanya diberi nilai 0.001 dikarenakan berdasarkan studi literatur nilai 0.001 adalah nilai yang baik untuk memulai titik untuk learning rate. Berangkat dari titik ini nanti akan di observasi untuk learning rate 0.001 - 0.05 dan juga learning rate 0.001 - 0.005.

Untuk pemilihan batch 1 juga didasari pada studi literatur bahwa titik terbaik untuk memulai pada batch sama dengan 1. sedangkan epochs pada 200 di dasari pada rentang epochs yang besar akan memudahkan melihat bagaimana prilaku dari akurasi selama epochs berjalan. sedangkan hidden unit dan banyak nya layer tidak ada standar yang baku kenapa memilih layer hidden unit sejumlah itu. Ini seperti tunning gitar mengandalkan intuisi, walaupun sebenarnya ada cara untuk tuning parameter secara matematis, namun untuk kali ini memang tidak memakai cara tersebut.

Setelah menentukan hyperparameter selanjutnya adalah mengimplementasikannya pada code.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.optimizers import RMSprop
from sklearn.model_selection import train_test_split
from sklearn.model_selection import StratifiedKFold
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
```

Baris-baris code ini adalah baris code untum mengimport beberapa library salah satunya tensor flow library yang kali digunakan untuk membangun model MLP, dan scikit learn untuk cross validation dan split data, sedangkan matplotlib dan seaborn untuk visualisasi data.

```
datas =
pd.read_csv("/home/khalifardy/Dokumen/ruang_kerja/code/kuliah/ML/Assignment_CLO1_pro
ject_base/prediksi_dropout/pre_processing_data/data_eda.csv")
```

datas variabel yang menyimpan data frame berdasarkan data yang dibaca pada direktori terpilih.

```
datas["Target"] = datas["Target"].map( {
"Dropout":1,
"Graduate":0,
"Enrolled":0
})
```

Kemudian pada baris code ini value pada kolom target dirubah valuenya kedalam bentuk integer dimana yang tadinya 3 kategori hanya menjadi 2 kategori. Untuk kategori Graduate dan Enrolled dijadikan satu class, karena pada dasarnya prediksi yang ingi dicapai adalah apakah seorang mahasiswa drop out atau tidak.

```
X = datas[datas.columns[:-1]].to_numpy()
Y = datas[datas.columns[-1]]
```

Pemisahan Data fitur yang disimpan dalam variabel X, data juga dirubah kedalam bentuk numpy array karena model pada tensor flow hanya menerima data berbentuk numpy array. Sedangkan untuk kolom target disimpan di dalam variabel Y.

```
dim = len(X)
models = tf.keras.Sequential(
[tf.keras.layers.Dense(units=dim, input_shape=(len(datas.columns[:-1]),)),
tf.keras.layers.Dense(512, activation='leaky_relu'),
tf.keras.layers.Dense(264, activation='leaky_relu'),
tf.keras.layers.Dense(128, activation='leaky_relu'),
tf.keras.layers.Dense(64, activation='leaky_relu'),
tf.keras.layers.Dense(32, activation='leaky_relu'),
tf.keras.layers.Dense(16, activation='leaky_relu'),
tf.keras.layers.Dense(8, activation='leaky_relu'),
tf.keras.layers.Dense(4, activation='leaky_relu'),
tf.keras.layers.Dense(2, activation='leaky_relu'),
tf.keras.layers.Dropout(0.2),
tf.keras.layers.Dense(1, activation='sigmoid')]
)
models.summary()
```

Pada implementasi ini mulai dibangun model dengan layer-layers nya, untuk layers input berisi perceptron sebnayak jumlah recordnya, dengan input shape sebayak jumlah kolom fitur x , 1 . Lalu berturut-turut hidden layer sebanyak 512 perceptron, 264 perceptron sampai 2 perceptron dengan fungsi aktivasi semuanya ‘leaky-relu’, pemilihan fungsi aktivasi leaky_relu dikarenakan fungsi yang mudah dan gradient bisa tetap terjaga. Lalu sebelum pada layer output terdapat layer dropout untuk menghindari overfitting. angka 0.2 adalah probabilitas suatu data akan dinonaktifkan pada suatu iterasi. Yang terakhir adalah layer output, dengan fungsi aktiasi nya sigmoid. Berhubung ini adalah kasus klasifikasi biner maka yang dipakai adalah fungsi aktivasi sigmoid.

```
epoch = 200
models.compile(
loss='binary_crossentropy',
optimizer=RMSprop(learning_rate=0.0007),
metrics=['accuracy']
)
history = models.fit(X,Y,validation_split=0.33,epochs=epoch,verbose=2)
```

Pada baris ini model sudah mulai dibangun dengan loss function binary crossentropy karena yang diklasifikasikan adalah tipe data kategorikal. optimizer yang digunakan adalah root mean square propagation dengan learning rate 0.0007, metrics yang diukur adalah akurasi. Lalu model di training dengan data validasi sebanyak 0.33 persen dan epochs sebanyak 200.

```
num_folds = 5
skf = StratifiedKFold(n_splits=num_folds, shuffle=True,random_state=22)

for index_train, index_test in skf.split(X,Y):
```

```
train_x, test_x = X[index_train],X[index_test]
train_y, test_y = Y[index_train], Y[index_test]
```

```
models.compile(
loss='binary_crossentropy',
optimizer=RMSprop(learning_rate=0.0007),
metrics=['accuracy']
)
```

```
models.fit(train_x,train_y,epochs=200,verbose=0)
scores = models.evaluate(test_x,test_y, verbose=0)
```

```
print("Validation accuracy: {:.2f}%".format(scores[1] * 100))
```

Baris code ini adalah pengimplementasian untuk cross validasi lalu setiap satu pasang data sudah selesai di train akan di cetak akurasinya. Karena ada 5 folds maka akan tercetak sebanyak 5 kali akurasi.

3.3.2. Implementasi LSTM

Pemilihan hyperparameter pada LSTM tidak berbeda jauh dengan pemilihan hyperparameter pada MLP. yang berbeda hanya pada input layer dan jumlah hidden layernya saja. Adapun hyperparameter pada LSTM bisa dilihat pada tabel berikut.

Parameter	Value
learning rate	0.001
batch	1
epochs	200
hidden unit	1
input first layer unit	64
Banyak nya layer	4

Table 3.3.2.1. Hyperparameter LSTM

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.optimizers import Adam,SGD, RMSprop
import pandas as pd
from sklearn.model_selection import train_test_split
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import StratifiedKFold
```

Code diatas mengimport library yang dibutuhkan tidak berbeda jauh dengan MLP, hanya ada penambahan LSTM sebagai model LSTM yang akan dibangun.

```
datas =
pd.read_csv("/home/khalifardy/Dokumen/ruang_kerja/code/kuliah/ML/Assignment_CLO1_pro
ject_base/prediksi_dropout/pre_processing_data/data_eda.csv")
X = datas[datas.columns[:-1]]
Y = datas['Target'].values
```

Membaca data pada direktori terpilih, lalu memisahkan data fitur dengan kolom target dan masing-masing disimpan pada variabel X dan Y.

```
label_mapping = {
"Dropout":0,
"Graduate":1,
"Enrolled":1
}

Y = [label_mapping[i] for i in Y]
```

sama seperti MLP sebelumnya value pada kolom target dirubah kedalam bentuk integer dan value graduated dan enrolled dijadikan satu kategori.

```
X_scaled = X.to_numpy()
X_scaled = X_scaled.reshape(X_scaled.shape[0],X_scaled.shape[1],1)
Y_scaled = np.array(Y)
```

Code diatas merubah X dan Y kedalam bentuk numpy array, lalu data fitur akan direshape sesuai dimensi nya namun ditambahkan satu dimensi, sebagai jumlah sequential , karena model LSTM hanya menerima data fitur dalam bentuk array 3 Dimensi, dimana dimensi terakhir adalah jumlah sequential.

```
model = Sequential()
model.add(LSTM(64,activation='leaky_relu',input_shape=(X_scaled.shape[1],X_scaled.sh
ape[2]),return_sequences=True))
#model.add(Dense(32, activation='relu'))
model.add(LSTM(32,activation='leaky_relu',return_sequences=False))
model.add(Dropout(0.2))
model.add(Dense(1, activation="sigmoid"))

optimizer = RMSprop(learning_rate=0.0001)
```

```
model.compile(loss='binary_crossentropy',optimizer=optimizer,metrics=['accuracy'])
history = model.fit(X_scaled,Y_scaled,validation_split=0.33,epochs=200,verbose=2)
```

Pada implementasi code ini , mulai dibangun model dengan input layer adalah LSTM cell dengan actiavsi leaky_relu dan sequensial bernilai true, karena sehabis input layer layer berikutnya adalah cell LSTM juga . Lalu layer berikutnya berisi 32 perceptron berupa cell LSTM dengan fungsi aktivasi leaky relu dan sequence False. Setelah itu Dropout untuk menghindari overfitting lalu layer output dengan fungsi aktivasi sigmoid. Loss function pada model ini binary crossentroppy dengan optimizer root mean squared propagation dan learning rate 0.0001.

Model juga ditraining dengan epochs sebanyak 200 dengan data dipisah untuk validasi sebanyak 33%.

Link github : *https://github.com/khalifardy/prediksi_dropout*

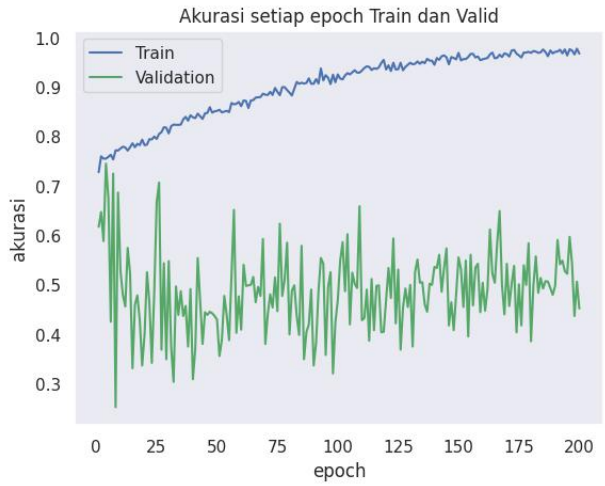
Bab 4

Evaluasi hasil

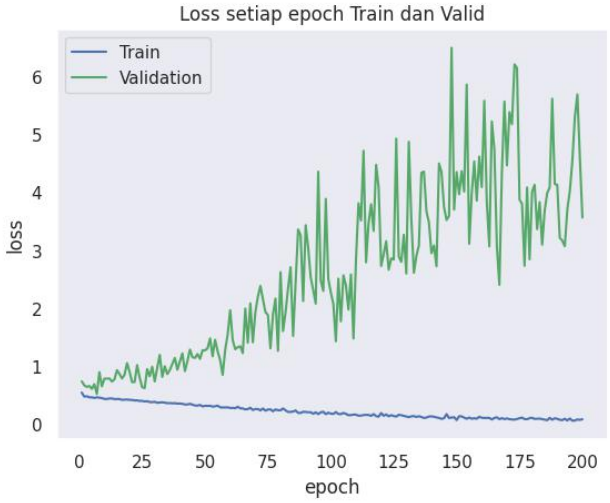
Pengukuran performansi model pada makalah ini menggunakan metrics akurasi. Sehingga dilakukan evaluasi model dengan menggunakan cross validation pada setiap model untuk didapatkan nilai rata-rata akurasi pada setiap model. Selain itu di evaluasi juga pada setiap rentang learning rate 0.0005 -0.001 dan 0.001 - 0.05 untuk didapatkan nilai learning rate untuk menghasilkan akurasi terbaik.

4.1. Hasil MLP

Pada MLP di nilai learning rate **0.001** didapatkan akurasi pada epoch 200 adalah **81.37 %** Namun jika dilihat pada histori training akan terlihat terdapat overfitting karena nilai akurasi pada train set jauh lebih tinggi daripada nilai akurasi pada validation set. Hal ini ditunjukan pada perbandingan akurasi tiap epoch dan loss tiap epoch train dan validation pada gambar dibawah ini.



Gambar 4.1. Akurasi setiap epoch train dan valid MLP



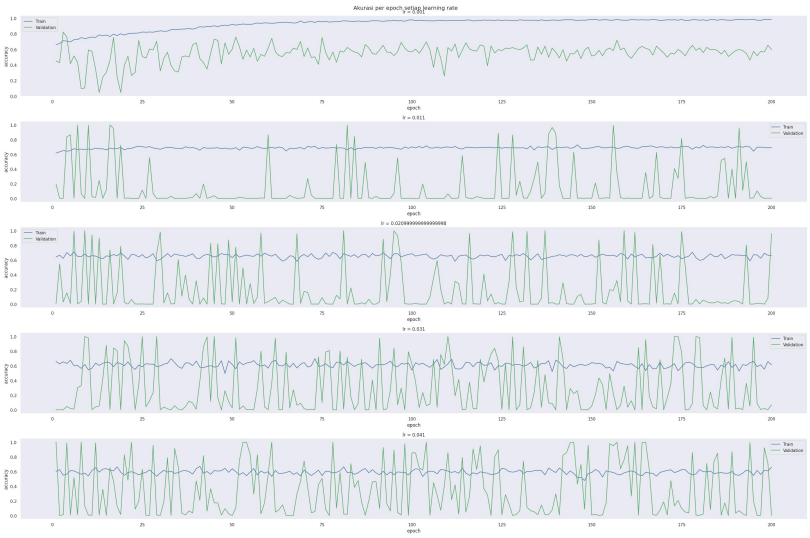
Gambar 4.2. Loss setiap epoch train dan valid MLP

Pada grafik diatas terlihat jelas bahwa akurasi maupun loss train tiap epoch cenderung stabil dan konvergen. Namun untuk validation baik akurasi dan loss cenderung tidak stabil dan tidak konvergen bahkan cenderung overfitting karena akurasi berada dibawah data train. Untuk mengatasi hal ini maka dilakukan cross validation untuk memastikan bagaimana sebenarnya prilaku model terhadap data baru. hasilnya akurasi rata-rata di dapatkan sebanyak **90.23 %** Hasil ini menunjukan bahwa model cukup baik untuk memprediksi data baru

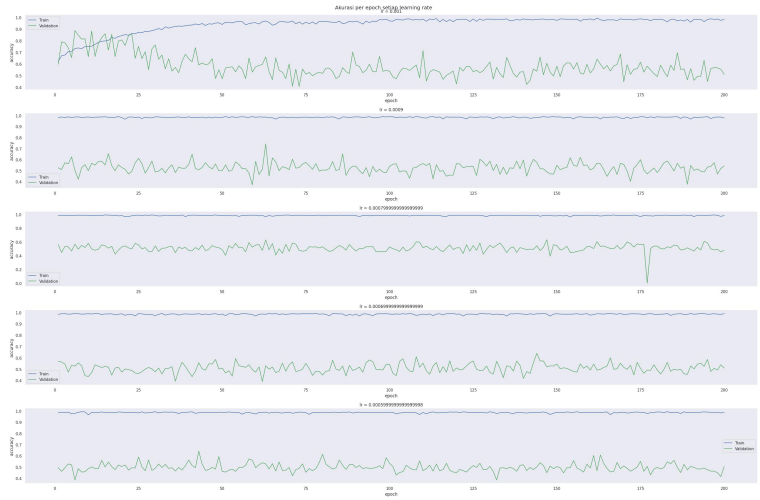
```
Validation accuracy: 82.95%
Validation accuracy: 87.50%
Validation accuracy: 95.95%
Validation accuracy: 92.78%
Validation accuracy: 94.19%
```

Gambar4.3. Validasi akurasi tiap fold MLP

Selain itu di lakukan juga evaluasi learning rate pada rentang 0.0005 -0.001 dan 0.001 - 0.05 . Hasilnya ditunjukan pada gambar-gambar dibawah ini.



Gambar 4.4. Akurasi rentang 0.001 - 0.05 MLP

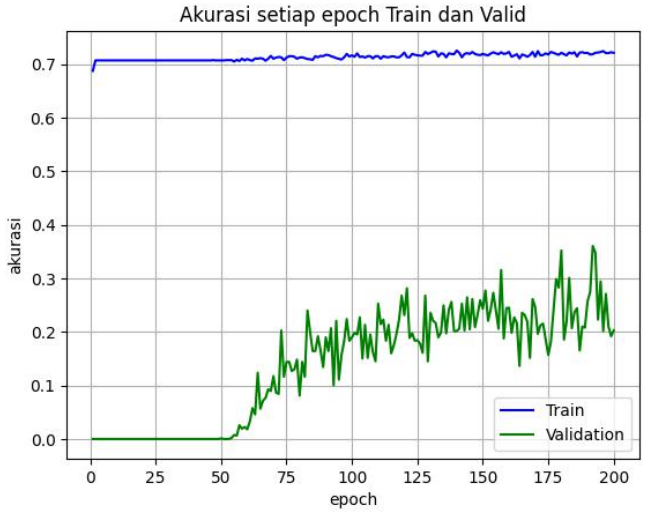


Gambar 4.5. Akurasi rentang 0.0005-0.001 MLP

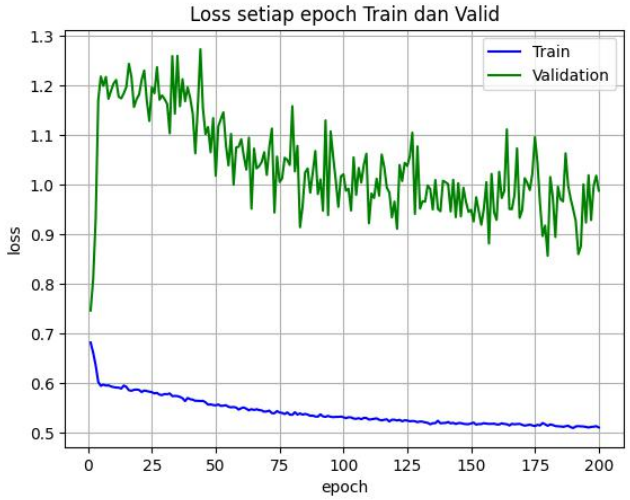
Dari hasil ini bisa dilihat memang karakterisitiknya masih sama bahwa model cenderung overfitting namun jika dilakukan cross validation di dapatkan bahwa untuk learning-rate pada rentang 0.0005 -0.001 memiliki tingkat akurasi yang lebih baik dibandingkan dengan learning rate pada rentang 0.001 - 0.05. Akurasi terbaik yang didapatkan ada pada learning rate 0.0005 dengan akurasi **99.68 %**

4.2. Hasil LSTM

LSTM pada learning rate 0.001 menghasilkan akurasi sebesar **75,6 %** . Namun sama seperti halnya MLP karakteristik modelnya cenderung overfitting seperti yang bisa dilihat pada gambar dibawah ini.



Gambar 4.6. Akurasi setiap epoch train dan valid LSTM



Gambar 4.7. Akurasi setiap epoch train dan valid LSTM

Seperti halnya pada MLP akurasi model valid pada LSTM cenderung tidak stabil dan divergen berkebalikan dengan model train nya yang konvergen dan stabil. Tidak berbeda jauh juga dengan loss nya.

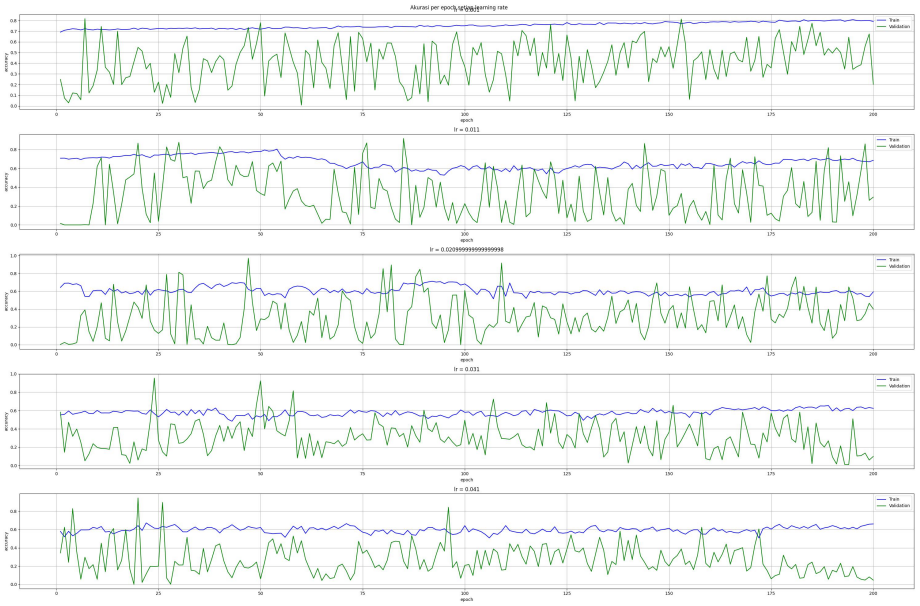
Treatment yang sama juga dilakukan pada LSTM yaitu melakukan cross validation untuk mengetahui bagaimana karakteristik model jika di berikan data baru

hasilya akurasi cukup stabil dengan rata-rata akurasi di **77.5 %** . Akurasi pada setiap fold bisa dilihat pada gambar dibawah ini.

```
Validation accuracy: 74.69%
Validation accuracy: 76.23%
Validation accuracy: 80.28%
Validation accuracy: 79.58%
Validation accuracy: 76.58%
```

Gambar 4.8. validasi akurasi cross validasi LSTM

Selain itu di lakukan juga evaluasi learning rate pada rentang 0.0005 -0.001 dan 0.001 - 0.05 . Hasilnya ditunjukan pada gambar-gambar dibawah ini.



Gambar 4.9. Akurasi rentang 0.001 - 0.05 LSTM

Gambar 4.10. Akurasi rentang 0.001 - 0.0005 LSTM

Seperti yang terlihat akurasi pada rentang 0.001 - 0.05 cenderung turun , sedangkan pada rentang 0.001 - 0.0005 memiliki akurasi yang lebih baik dengan akurasi terbaik ada pada learning rate 0.0005 dengan akurasi **75.34 %.**

4.3. Kesimpulan

Berdasarkan pada hasil evaluasi model dapat disimpulkan pemakaian model terbaik untuk kasus prediksi dropout mahasiswa adalah memakai model MLP dengan learning rate 0.0005, karena memiliki akurasi **99.68 %** jika dibandingkan dengan model LSTM yang memiliki akurasi terbaik pada **75.34 %.**

LSTM yang memiliki akurasi lebih rendah dibandingkan dengan MLP sudah bisa diprediksi sebelumnya, hal ini disebabkan karakteristik data yang bukan tipe data sekuensial yang berdasarkan waktu-waktu sebelumnya.

Namun sebenarnya hal ini bisa dipelajari lebih jauh lagi , karena penyebabnya bisa jadi karena pemilihan hyperparameter yang kurang optimal atau dataset yang kurang di preporcessing dengan baik, hal ini ditandai pada karakteristik model yang sebelumnya di train cenderung untuk overfitting.

Untuk itu untuk saran pada penelitian kedepan bisa dilakukan tuning hyperparameter yang optimal sehingga model yang dilatih benar-benar model yang memilki performansi terbaik.

Daftar Pustaka

Prof. Dr. Suyanto, S.T, M.sc.2022. Machine learning tingkat dasar dan lanjut. Bandung. Penerbit Informatika

Dr. Suyanto. 2019. Data Mining untuk klasifikasi dan klasterisasi data. Bandung. Penerbit Informatika

PPT Pembelajaran Mesin Universitas Telkom SSD-ADF-ATW-SUY

https://towardsdatascience.com/hyperparameters-in-deep-learning-927f7b2084dd

https://medium.com/sysinfo/multi-layer-perceptron-6ccaace0dcc8

https://medium.com/@samuelsena/pengenalan-deep-learning-part-3-backpropagation-algorithm-720be9a5fbb8

https://pub.towardsai.net/introduction-to-deep-learning-part-2-rnns-and-ltsm-fc65c230713d

Lampiran

link github : *https://github.com/khalifardy/prediksi_dropout*

link drive : *https://drive.google.com/drive/folders/1Y8Y4KLrE4s-9ob0RMLJrAAwHuvGAqfuC?usp=sharing*