LAPORAN CASE BASED-1 MACHINE LEARNING

Oleh :

Muhammad Daffa' Ibrahim (1301204051)

PROGRAM STUDI S1 INFORMATIKA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM
2022

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Deep learning merupakan sub bidang *machine learning* yang algoritmanya terinspirasi dari struktur otak manusia. Saat ini, teknik *deep learning* sangat populer di kalangan praktisi data dan menarik perhatian banyak pihak. Hal ini karena teknologi *deep learning* telah diterapkan dalam berbagai produk berteknologi tinggi seperti self-driving car. Selain itu, ia juga ada di balik produk dan layanan yang kita gunakan sehari-hari. Contohnya antara lain, asisten digital, Google Translate, dan *voice-activated device* (perangkat cerdas yang bisa diaktifkan dengan suara). Beberapa algoritma terkait *Deep learning* yang dipelajari pada perkuliahan diantaranya adalah *Artificial Neural Network* (ANN), *Multi Layer Perceptron* (MLP), *Recurrent Neural Network* (RNN), *Long-short Term Memory* (LSTM), dan *Convolutional Neural Network* (CNN).

1.2 Spesifikasi Tugas

Diberikan file trial.xlsx yang berisi dataset untuk problem klasifikasi biner (binary classification). Dataset tersebut menggambarkan record dari beberapa perusahaan yang dicurigai. Setiap record atau baris data dalam dataset tersebut secara umum terdiri dari 18 atribut termasuk atribut yang menjadi target klasifikasi biner. Fitur input terdiri dari nilai-nilai float dalam range tertentu untuk setiap fitur. Sedangkan atribut target bernilai biner (0 atau 1).

```
     Sector_score LOCATION_ID  PARA_A  SCORE_A  PARA_B  SCORE_B  TOTAL  \
0            3.89          23    4.18        6    2.50        2   6.68
1            3.89           6    0.00        2    4.83        2   4.83
2            3.89           6    0.51        2    0.23        2   0.74
3            3.89           6    0.00        2   10.80        6  10.80
4            3.89           6    0.00        2    0.08        2   0.08
..            ...         ...     ...      ...     ...      ...    ...
771         55.57           9    0.49        2    0.40        2   0.89
772         55.57          16    0.47        2    0.37        2   0.84
773         55.57          14    0.24        2    0.04        2   0.28
774         55.57          18    0.20        2    0.00        2   0.20
775         55.57          15    0.00        2    0.00        2   0.00

     numbers  Marks  Money_Value  MONEY_Marks  District  Loss  LOSS_SCORE  \
0        5.0      2         3.38            2         2     0           2
1        5.0      2         0.94            2         2     0           2
2        5.0      2         0.00            2         2     0           2
3        6.0      6        11.75            6         2     0           2
4        5.0      2         0.00            2         2     0           2
..       ...    ...          ...          ...       ...   ...         ...
771      5.0      2         0.00            2         2     0           2
772      5.0      2         0.00            2         2     0           2
773      5.0      2         0.00            2         2     0           2
774      5.0      2         0.00            2         2     0           2
775      5.0      2         0.32            2         2     0           2

     History  History_score  Score  Risk
0          0              2    2.4     1
1          0              2    2.0     0
2          0              2    2.0     0
3          0              2    4.4     1
4          0              2    2.0     0
..       ...            ...    ...   ...
771        0              2    2.0     0
772        0              2    2.0     0
773        0              2    2.0     0
774        0              2    2.0     0
775        0              2    2.0     0

[776 rows x 18 columns]
```

1.3 Ikhtisar Kumpulan Data yang Dipilih

1.3.1 Missing value

Terdapat sedikit missing value pada dataset trial.xlsx ini, tepat nya pada atribut Money_Value sehingga saya melakukan assign ulang pada missing value tersebut dengan nilai rata-rata pada atribut Money_Value.

```
Sector_score     0
LOCATION_ID      0
PARA_A           0
SCORE_A          0
PARA_B           0
SCORE_B          0
TOTAL            0
numbers          0
Marks            0
Money_Value      1
MONEY_Marks      0
District         0
Loss             0
LOSS_SCORE       0
History          0
History_score    0
Score            0
Risk             0
dtype: int64
```

1.3.2 Data berbentuk String

Terdapat beberapa data yang masih berbentuk string, tepatnya pada atribut LOCATION_ID sehingga saya menghilangkan data tersebut, sekaligus mengubah semua data menjadi tipe data float.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 773 entries, 0 to 775
Data columns (total 18 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Sector_score   773 non-null    float64
 1   LOCATION_ID    773 non-null    float64
 2   PARA_A         773 non-null    float64
 3   SCORE_A        773 non-null    float64
 4   PARA_B         773 non-null    float64
 5   SCORE_B        773 non-null    float64
 6   TOTAL          773 non-null    float64
 7   numbers        773 non-null    float64
 8   Marks          773 non-null    float64
 9   Money_Value    772 non-null    float64
 10  MONEY_Marks    773 non-null    float64
 11  District       773 non-null    float64
 12  Loss           773 non-null    float64
 13  LOSS_SCORE     773 non-null    float64
 14  History        773 non-null    float64
 15  History_score  773 non-null    float64
 16  Score          773 non-null    float64
 17  Risk           773 non-null    float64
dtypes: float64(18)
memory usage: 114.7 KB
```

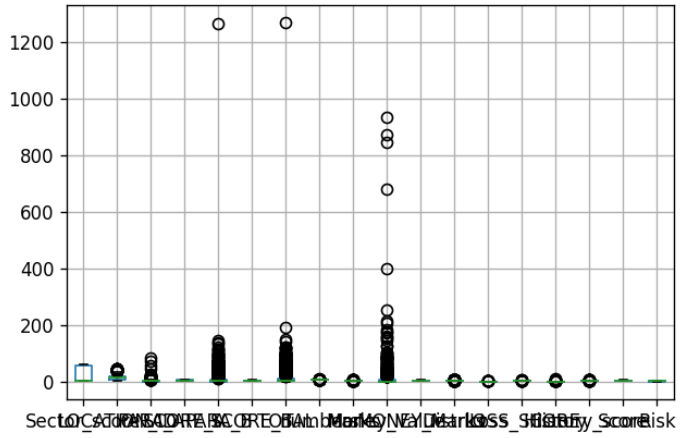
1.3.3 Data Duplikat

Terdapat beberapa data yang menduplikat.

```
0      False
1      False
2      False
3      False
4      False
       ...
771     True
772    False
773    False
774    False
775    False
Length: 773, dtype: bool
```

1.3.4 Data Outlier

Terdapat beberapa data yang berada diluar batasnya. jika divisualisasikan menggunakan boxplot mungkin lebih terlihat jelas.

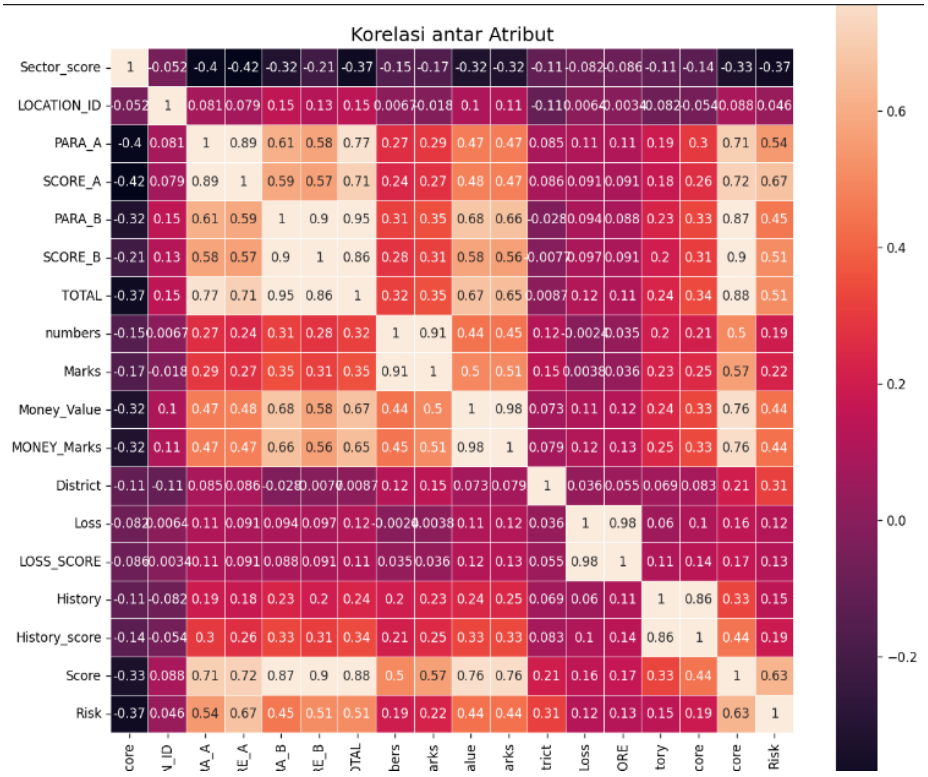


1.3.5 Atribut Data

Dataset Trial.xlsx ini memiliki banyak sekali atribut, terdapat 18 atribut dari data set ini yang mana akan menyulitkan proses pemodelan serta menjadikan data menjadi kurang bagus untuk dilakukan proses pemodelan. Oleh karenanya kita memerlukan proses reduksi dimensi untuk mengubah jumlah atribut dari dataset menjadi lebih sedikit

1.3.6 Korelasi Data

Mencari nilai dari korelasi antar atribut dari dataset.



Atribut yang memiliki korelasi paling tinggi dengan atribut target (risk) adalah score, akan tetapi dikarenakan saya tidak dapat memahami arti setiap atributnya, maka proses reduksi dimensi akan dilakukan menggunakan Principal Component Analysis (PCA).

BAB 2
PRA-PEMROSESAN DATA

2.1 Missing Value

Hal pertama dalam pra-pemrosesan data ini adalah menghapus missing value yang terdapat pada dataset ini.

```
Sector_score     0
LOCATION_ID      0
PARA_A           0
SCORE_A          0
PARA_B           0
SCORE_B          0
TOTAL            0
numbers          0
Marks            0
Money_Value      1
MONEY_Marks      0
District         0
Loss             0
LOSS_SCORE       0
History          0
History_score    0
Score            0
Risk             0
dtype: int64
```

Terdapat 1 buah missing value yang terdapat pada atribut Money_Value. sehingga saya akan mengganti missing value tersebut dengan nilai rata-rata dari atribut Money_Value.

```
df['Money_Value'].fillna(df['Money_Value'].mean(), inplace = True)
```

setelah missing value tersebut diubah, maka tidak ada lagi missing value pada dataset ini.

```
df.isnull().sum()
```

```
Sector_score     0
LOCATION_ID      0
PARA_A           0
SCORE_A          0
PARA_B           0
SCORE_B          0
TOTAL            0
numbers          0
Marks            0
Money_Value      0
MONEY_Marks      0
District         0
Loss             0
LOSS_SCORE       0
History          0
History_score    0
Score            0
Risk             0
dtype: int64
```

2.2 Data Bertipe String

Terdapat beberapa data yang masih berbentuk String. Data tersebut berasal dari atribut LOCATION_ID yang menandakan lokasi perusahaan tersebut. agar data dapat diproses pada pemodelan, data harus berbentuk angka. Juga untuk mempermudah proses pemodelan, saya mengubah tipe data menjadi float untuk semua data pata dataset.

```
df = df[(df.LOCATION_ID != 'LOHARU')]
df = df[(df.LOCATION_ID != 'NUH')]
df = df[(df.LOCATION_ID != 'SAFIDON')]
df = df.astype(float)
```

Beberapa data seperti ‘LOHARU’, ‘NUH’, dan ‘SAFIDON’ masih berbentuk String, sehingga saya drop data tersebut sekaligus mengubah seluruh data menjadi bertipe float.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 773 entries, 0 to 775
Data columns (total 18 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Sector_score   773 non-null    float64
 1   LOCATION_ID    773 non-null    float64
 2   PARA_A         773 non-null    float64
 3   SCORE_A        773 non-null    float64
 4   PARA_B         773 non-null    float64
 5   SCORE_B        773 non-null    float64
 6   TOTAL          773 non-null    float64
 7   numbers        773 non-null    float64
 8   Marks          773 non-null    float64
 9   Money_Value    772 non-null    float64
 10  MONEY_Marks    773 non-null    float64
 11  District       773 non-null    float64
 12  Loss           773 non-null    float64
 13  LOSS_SCORE     773 non-null    float64
 14  History        773 non-null    float64
 15  History_score  773 non-null    float64
 16  Score          773 non-null    float64
 17  Risk           773 non-null    float64
dtypes: float64(18)
memory usage: 114.7 KB
```

2.3 Data Duplikat

Terdapat beberapa data record yang menduplikat atau sama dengan data record lainnya pada dataset ini.

```
0      False
1      False
2      False
3      False
4      False
       ...
771     True
772    False
773    False
774    False
775    False
Length: 773, dtype: bool
```

Data yang menduplikat tadi kemudian akan saya hilangkan atau saya drop sehingga tidak terdapat lagi data record yang sama dengan data record lainnya pada dataset.

```
df.duplicated()
df= df.drop_duplicates()
```

```
df.duplicated()
```

```
0      False
1      False
2      False
3      False
4      False
       ...
770    False
772    False
773    False
774    False
775    False
Length: 760, dtype: bool
```

2.4 Handling Outlier

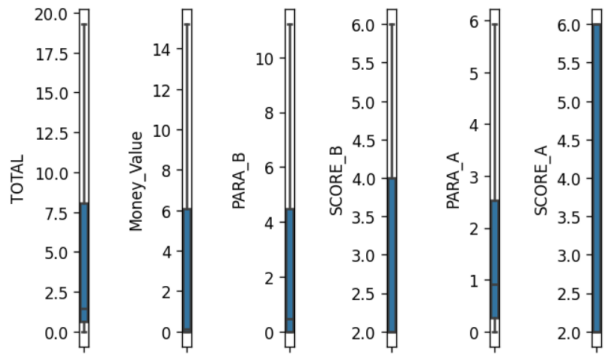
Terdapat beberapa outlier data yang dapat menyebabkan dataset kurang bagus untuk dilakukan pemodelan. Outlier tersebut ditangani dengan memberi batasan terhadap outlier. Batas nilai maksimum yang digunakan adalah nilai (q3 + (interquartile * 1.5)) sedangkan batas nilai minimum yang digunakan adalah nilai (q1 - (interquartile * 1.5)). Jika terdapat nilai yang melebihi batas maksimum, maka nilai tersebut akan di assign dengan nilai pada batas maksimum, Jika terdapat nilai yang lebih kecil batas minimum, maka nilai tersebut akan di assign dengan nilai pada batas minimum.

```
# handling outlier data train

# menghitung jarak interquartile
def interquartile(data,x):
  q1 = (data[x]).quantile(0.25)
  q3 = (data[x]).quantile(0.75)
  iqr = q3 - q1
  maximum = q3 + (1.5 *iqr)
  minimum = q1 - (1.5 *iqr)
  return maximum,minimum

  # menggantikan value outliners dengan hasil dari perhitungan jarak interquartile
def sub_outliners(data,x,maximum,minimum):
  more_than = (data[x] > maximum)
  less_than = (data[x] < minimum)
  print('more_than: ',more_than,' | less_than: ',less_than)
  data[x] = data[x].mask(more_than, maximum,axis=0)
  data[x] = data[x].mask(less_than, minimum,axis=0)
  return data
```

sehingga data outlier dapat ditangani dan tidak lagi terdapat data outlier.



2.5 Normalisasi Data

Dataset yang digunakan memiliki nilai data yang beragam, terdapat data dengan nilai data yang berjumlah besar (> 20), terdapat juga data dengan nilai data yang kecil (< 1). Oleh Karena itu, untuk memudahkan proses eksekusi pemodelan, perlu adanya normalisasi data untuk mengubah nilai dari data menjadi data berskala 0 - 1.

```
normalizedData = (data-np.min(df))/(np.max(df)-np.min(df))
print(normalizedData)
```

sehingga nilai dari setiap datanya akan berskala dari range 0 sampai dengan .

```
Sector_score  LOCATION_ID    PARA_A  SCORE_A    PARA_B  SCORE_B  \
    0.035172     0.511628  0.705336      1.0  0.223090      0.0
    0.035172     0.116279  0.000000      0.0  0.431009      0.0
    0.035172     0.116279  0.086058      0.0  0.020524      0.0
    0.035172     0.116279  0.000000      0.0  0.963748      1.0
    0.035172     0.116279  0.000000      0.0  0.007139      0.0
         ...          ...       ...      ...       ...      ...
    0.926207     0.395349  0.126556      0.0  0.040156      0.0
    0.926207     0.348837  0.079308      0.0  0.033017      0.0
    0.926207     0.302326  0.040498      0.0  0.003569      0.0
    0.926207     0.395349  0.033748      0.0  0.000000      0.0
    0.926207     0.325581  0.000000      0.0  0.000000      0.0

   TOTAL  numbers  Marks  Money_Value  MONEY_Marks  District  Loss  \
0.346383     0.00    0.0     0.222368          0.0       0.0   0.0
0.250454     0.00    0.0     0.061842          0.0       0.0   0.0
0.038372     0.00    0.0     0.000000          0.0       0.0   0.0
0.560021     0.25    1.0     0.773026          1.0       0.0   0.0
0.004148     0.00    0.0     0.000000          0.0       0.0   0.0
     ...      ...    ...          ...          ...       ...   ...
0.062225     0.00    0.0     0.000000          0.0       0.0   0.0
0.043557     0.00    0.0     0.000000          0.0       0.0   0.0
0.014519     0.00    0.0     0.000000          0.0       0.0   0.0
0.010371     0.00    0.0     0.000000          0.0       0.0   0.0
0.000000     0.00    0.0     0.021053          0.0       0.0   0.0
```

2.6 Reduksi Dimensi

Dataset Trial.xlsx memiliki cukup banyak atribut, yaitu sebanyak 18 atribut yang berbeda. Jumlah atribut yang banyak akan menyulitkan proses pemodelan dan memperburuk kualitas dataset. oleh karena itu dibutuhkan proses reduksi dimensi untuk membuat atribut menjadi lebih sedikit. Metode proses reduksi dimensi yang digunakan adalah metode Principal Component Analysis (PCA).

Principal Component Analysis (PCA) adalah salah satu metode reduksi dimensi pada machine learning. PCA akan memilih variabel-variabel yang mampu menjelaskan sebagian besar variabilitas data. PCA mengurangi dimensi dengan membentuk variabel-variabel baru yang disebut Principal Components.

```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
fit_pca = pca.fit_transform(normalizedData)
data_fit = pd.DataFrame(data = fit_pca, columns = ['PCA_1', 'PCA_2'])
data_fit['Data_y'] = data_y
print(data_fit)
```

Atribut yang berjumlah 18 kemudian direduksi menjadi 2 parameter baru yaitu PCA_1 dan PCA_2, tidak lupa juga kita tambahkan atribut yang menjadi target, pada kasus ini data target tersebut adalah data_y.

```
        PCA_1     PCA_2  Data_y
0   -1.006564 -0.364498     1.0
1   -1.025972 -0.642324     0.0
2   -1.474288 -0.533836     0.0
3   -0.098767 -0.138783     1.0
4   -1.506412 -0.531609     0.0
..        ...       ...     ...
755 -1.456209 -0.546389     1.0
756 -1.473595 -0.545084     0.0
757 -1.513441 -0.534607     1.0
758 -1.518740 -0.533762     0.0
759 -1.518502 -0.515050     0.0

[760 rows x 3 columns]
```

2.7 Validasi data

Proses validasi data yang digunakan menggunakan metode *Holdout Validation* yang mana membagi data train sebanyak 75% dari data keseluruhan dan data uji sebanyak 25% dari data keseluruhan.

```
latih = int((0.75* len(data_fit)))
dataLatih = data_fit[:int(latih)]
dataLatih = dataLatih.astype(float)
dataLatih.fillna(0,inplace=True)
dataUji = data_fit[int(latih): len(data_fit)]
```

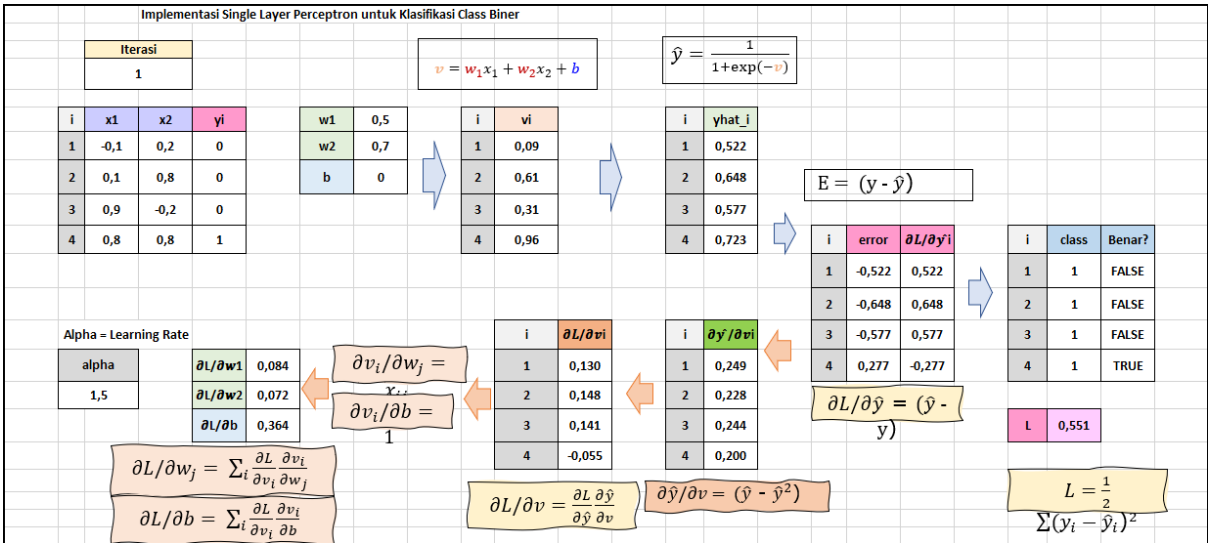
BAB 3
ALGORITMA YANG DIPILIH

Algoritma *supervised learning* yang digunakan adalah algoritma *Artificial Neural Network* atau ANN. ANN merupakan model penalaran yang didasarkan pada otak manusia. ANN terdiri dari sejumlah prosesor sangat sederhana dan saling berhubungan yang disebut neuron. Neuron yang terhubung dengan pembobotan (weight) melewati sinyal dari neuron satu ke neuron yang lain.

Pada kasus ini saya hanya akan menggunakan satu buah layer atau *Single Layer Perceptron* karena tugas yang diberikan adalah melakukan prediksi untuk 2 buah kategori yaitu antara 1 dan 0 atau biasa dikenal dengan *binary classification.*

3.1 Single Layer Perceptron

Secara konsep, proses pemodelan menggunakan *Single Layer Perceptron* terbagi menjadi 2 tahap, yaitu *forward pass* dan *backward pass* proses ini disebut dengan *BackPropagation*. *forward pass* merupakan proses yang terjadi dari belakang ke depan untuk melakukan proses prediksi terhadap data. Sedangkan *backward pass* merupakan proses dari depan ke belakang untuk menentukan hyper parameter yang tepat untuk meningkatkan nilai akurasi pada setiap iterasinya.



Forward pass ditandai dengan panah berwarna biru sedangkan *backward pass* ditandai dengan panah berwarna oranye.

3.2 Modelling *Forward Pass*

3.2.1 Menentukan nilai W dan B

Tahap awal dari *single layer perceptron* ini adalah melakukan assign nilai W dan B secara random (nilai W dan B yang digunakan lebih baik bernilai rendah, antara 0-1).

```
w1 = 0.3
w2 = 0.2
b = 0.1
```

3.2.2 Menentukan nilai V

Tahap selanjutnya adalah menentukan nilai v yang didapat dari rumus berikut:

$v=w1\ x1+w2\ x2+b$

nilai W1,W2, dan b akan diassign pada persamaan tersebut, nilai x1 dan x2 didapat dari nilai pada tiap atribut yang dimiliki data

```
def vix(w1,w2,b,x1,x2):
  target = (w1*x1) + (w2*x2) + b
  return target
```

3.2.3 Menentukan nilai ŷ

Setelah menemukan nilai v, maka selanjutnya adalah menentukan nilai ŷ yang didapat dari rumus berikut:

$\hat{y}= 1/(1+\exp(-v))$

```
def yhat_i(vi):
  target = 1/(1+(mp.exp(-vi)))
  return target
```

3.2.4 Melakukan prediksi data

Proses prediksi data dilakukan dengan menghitung apakah nilai ŷ lebih besar dari 0.5, jika lebih besar maka akan mengoutputkan nilai 1, jika lebih kecil akan mengoutputkan nilai 0.

```
def pred(yhat_i):
  if yhat_i > 0.5:
    target = 1
  else:
    target = 0
  return target
```

3.3 Modeling *Backward Pass*

3.3.1 Menghitung nilai ∂L/∂ŷ

nilai ∂L/∂ŷ didapatkan dari proses perhitungan sebagai berikut:

∂L/∂ŷ= (ŷ - y)

```
def DlDy(yhat_i, y):
  target = yhat_i - y
  return target
```

3.3.2 Menghitung nilai ∂ŷ/∂v

nilai ∂ŷ/∂v didapatkan dari proses perhitungan sebagai berikut:

∂ŷ/∂v = (ŷ- ŷ^2)

```
def DyDvi(yhat_i):
  target = yhat_i - (yhat_i**2)
  return target
```

3.3.3 Menghitung nilai ∂*L*/∂*v*i

nilai ∂*L*/∂*v*i didapatkan dari proses perkalian antara ∂L/∂ŷ dengan ∂ŷ/∂v

∂L/∂v = ∂L/(∂ŷ) * (∂ŷ)/∂v

```
def DlDvi(yhat_i, y):
  target = DlDy(yhat_i, y) * DyDvi(yhat_i)
  return target
```

3.3.4 Menentukan gradien w1 dan w2

nilai gradient didapatkan dari penjumlahan seluruh data dari hasil perkalian ∂*L*/∂*v*i dengan nilai data pada atributnya.

∂L/∂wj= ∑i 〖∂L/(∂vi) * (∂vi)/(∂wj)〗

```
def DlDw(yhat_i, y, x1):
  target = DlDvi(yhat_i,y) * x1
  return target
```

```
dldw1 = DlDw(yi, dataLatih.Data_y[i], dataLatih.PCA_1[i])
dldw2 = DlDw(yi, dataLatih.Data_y[i], dataLatih.PCA_2[i])
dw1 += dldw1
dw2 += dldw2
```

dw1 merupakan gradien untuk nilai w1 dan dw2 merupakan nilai gradien untuk nilai w2.

3.3.5 Menentukan gradien b

nilai gradient didapatkan dari penjumlahan seluruh nilai $\partial L/\partial v$i.

$$\partial L/\partial b = \sum i \; [\![\partial L/(\partial vi) * (\partial vi)/\partial b]\!]$$

dengan nilai $(\partial vi)/\partial b = 1$

```
dldv = DlDvi(yi,dataLatih.Data_y[i])
dldv1 = dldv1 + dldv
```

dldv1 merupakan jumlah dari keseluruhan nilai dldv.

3.3.6 Update nilai w1,w2, dan b

Update nilai adalah dengan:

*bobot baru = bobot lama - alpha * gradien*

yang mana alpha merupakan hyper parameter atau *learning rate* dari pemodelan tersebut.

```
def update(w,alpha, dw):
  w = w - (alpha*dw)
  return w
```

```
w1 = update(w1,alpha,dw1)
w2 = update(w2,alpha,dw2)
b  = update(b,alpha,dldv1)
```

BAB 4
IMPLEMENTASI DAN ANALISIS

Single Layer Perceptron

Pecobaan ke-1
Menggunakan nilai learning rate sebesar 1

```
#TRAIN DATA_TRAIN

#Hyper Parameter
w1 = 1
w2 = 1
b = 0.1
alpha = 1

#Forward Pass
j = 0
while (j < 100):
  array_check = []
  for i in range(len(dataLatih)):
    x1 = dataLatih.PCA_1[i]
    x2 = dataLatih.PCA_2[i]
    v = vix(w1,w2,b,x1,x2)
    yi = yhat_i(v)
    dldy = DlDy(yi, dataLatih.Data_y[i])
    dydvi = DyDvi(yi)
    check = pred(yi)
    array_check.append(check)
  acc = accuracy(dataLatih.Data_y, array_check)
  print("acc : ", acc)
```

```
#Backward Pass
  dldv1 = 0
  dw1 = 0
  dw2 = 0
  for i in range(len(dataLatih)):
    x1 = dataLatih.PCA_1[i]
    x2 = dataLatih.PCA_2[i]
    v = vix(w1,w2,b,x1,x2)
    yi = yhat_i(v)
    dldv = DlDvi(yi,dataLatih.Data_y[i])
    dldv1 = dldv1 + dldv
    dldw1 = DlDw(yi, dataLatih.Data_y[i], dataLatih.PCA_1[i])
    dldw2 = DlDw(yi, dataLatih.Data_y[i], dataLatih.PCA_2[i])
    dw1 += dldw1
    dw2 += dldw2

  w1 = update(w1,alpha,dw1)
  w2 = update(w2,alpha,dw2)
  b  = update(b,alpha,dldv1)
  j = j +1
```

Hasil, akurasi = 52 %

```
Confusion Matrix
[[  4 166]
 [108 292]]
Classification Report
              precision    recall  f1-score   support

         0.0       0.04      0.02      0.03       170
         1.0       0.64      0.73      0.68       400

    accuracy                           0.52       570
   macro avg       0.34      0.38      0.35       570
weighted avg       0.46      0.52      0.49       570
```

Percobaan ke-2
Menggunakan nilai learning rate sebesar 0.5

```
#TRAIN DATA_TRAIN

#Hyper Parameter
w1 = 1
w2 = 1
b = 0.1
alpha = 0.5

#Forward Pass
j = 0
while (j < 100):
  array_check = []
  for i in range(len(dataLatih)):
    x1 = dataLatih.PCA_1[i]
    x2 = dataLatih.PCA_2[i]
    v = vix(w1,w2,b,x1,x2)
    yi = yhat_i(v)
    dldy = DlDy(yi, dataLatih.Data_y[i])
    dydvi = DyDvi(yi)
    check = pred(yi)
    array_check.append(check)
  acc = accuracy(dataLatih.Data_y, array_check)
  print("acc : ", acc)
```

```
#Backward Pass
  dldv1 = 0
  dw1 = 0
  dw2 = 0
  for i in range(len(dataLatih)):
    x1 = dataLatih.PCA_1[i]
    x2 = dataLatih.PCA_2[i]
    v = vix(w1,w2,b,x1,x2)
    yi = yhat_i(v)
    dldv = DlDvi(yi,dataLatih.Data_y[i])
    dldv1 = dldv1 + dldv
    dldw1 = DlDw(yi, dataLatih.Data_y[i], dataLatih.PCA_1[i])
    dldw2 = DlDw(yi, dataLatih.Data_y[i], dataLatih.PCA_2[i])
    dw1 += dldw1
    dw2 += dldw2

  w1 = update(w1,alpha,dw1)
  w2 = update(w2,alpha,dw2)
  b  = update(b,alpha,dldv1)
  j = j +1
```

Hasil akurasi = 67%

```
Confusion Matrix
[[  3 167]
 [ 20 380]]
Classification Report
              precision    recall  f1-score   support

         0.0       0.13      0.02      0.03       170
         1.0       0.69      0.95      0.80       400

    accuracy                           0.67       570
   macro avg       0.41      0.48      0.42       570
weighted avg       0.53      0.67      0.57       570
```

Percobaan ke-3
Menggunakan nilai learning rate sebesar 0.05

```
#TRAIN DATA_TRAIN

#Hyper Parameter
w1 = 1
w2 = 1
b = 0.1
alpha = 0.05

#Forward Pass
j = 0
while (j < 100):
  array_check = []
  for i in range(len(dataLatih)):
    x1 = dataLatih.PCA_1[i]
    x2 = dataLatih.PCA_2[i]
    v = vix(w1,w2,b,x1,x2)
    yi = yhat_i(v)
    dldy = DlDy(yi, dataLatih.Data_y[i])
    dydvi = DyDvi(yi)
    check = pred(yi)
    array_check.append(check)
  acc = accuracy(dataLatih.Data_y, array_check)
  print("acc : ", acc)
```

```
#Backward Pass
  dldv1 = 0
  dw1 = 0
  dw2 = 0
  for i in range(len(dataLatih)):
    x1 = dataLatih.PCA_1[i]
    x2 = dataLatih.PCA_2[i]
    v = vix(w1,w2,b,x1,x2)
    yi = yhat_i(v)
    dldv = DlDvi(yi,dataLatih.Data_y[i])
    dldv1 = dldv1 + dldv
    dldw1 = DlDw(yi, dataLatih.Data_y[i], dataLatih.PCA_1[i])
    dldw2 = DlDw(yi, dataLatih.Data_y[i], dataLatih.PCA_2[i])
    dw1 += dldw1
    dw2 += dldw2

  w1 = update(w1,alpha,dw1)
  w2 = update(w2,alpha,dw2)
  b  = update(b,alpha,dldv1)
  j = j +1
```

Hasil akurasi = 70%

```
Confusion Matrix
[[  0 170]
 [  0 400]]
Classification Report
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00       170
         1.0       0.70      1.00      0.82       400

    accuracy                           0.70       570
   macro avg       0.35      0.50      0.41       570
weighted avg       0.49      0.70      0.58       570
```

Percobaan pada DataTest

Setelah melakukan percobaan, learning rate yang baik adalah sebesar 0.05, kita coba melakukan proses modeling pada data test dengan menerapkan learning rate sebesar 0.05.

```
#Save nilai W1,W2,dan B terbaik
W1_global = w1
W2_global = w2
B_global = b


#DATA TEST
i = 570
j = 0
array_check = []
while(j < len(dataUji)):
    x1 = dataUji.PCA_1[i]
    x2 = dataUji.PCA_2[i]
    v = vix(W1_global,W2_global,B_global,x1,x2)
    yi = yhat_i(v)
    dldy = DlDy(yi, dataUji.Data_y[i])
    dydvi = DyDvi(yi)
    check = pred(yi)
    array_check.append(check)
    i = i + 1
    j = j + 1
print(array_check)
```

Hasil akurasi dari pemodelan data test ini sebesar 56%

```
Confusion Matrix
[[95 13]
 [70 12]]
Classification Report
              precision    recall  f1-score   support

         0.0       0.58      0.88      0.70       108
         1.0       0.48      0.15      0.22        82

    accuracy                           0.56       190
   macro avg       0.53      0.51      0.46       190
weighted avg       0.53      0.56      0.49       190
```

Multi Layer Perceptron
Hidden Layer yang digunakan adalah sebanyak 5 layer

Percobaan ke-1
Menggunakan nilai learning rate sebesar 1

```
import tensorflow as tf

model = tf.keras.Sequential (
    [
        tf.keras.layers.Dense(128, activation = 'relu'),
        tf.keras.layers.Dense(64),
        tf.keras.layers.Dense(32),
        tf.keras.layers.Dense(16),
        tf.keras.layers.Dense(4),
        tf.keras.layers.Dense(2),
        tf.keras.layers.Dense(1, activation = 'sigmoid'),
    ]
)

model.compile(loss = 'binary_crossentropy', optimizer = tf.optimizers.Adam(learning_rate=1), metrics = ['accuracy', 'Precision', 'Recall'])
```

```
target = 0.9
class callbacks(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs = None):
        if logs.get('accuracy') >= target:
            print('\nFor Epoch', epoch, '\nAkurasi telah mencapai = %2.2f%%' %(logs['accuracy']*100), 'proses training selesai.')
            self.model.stop_training = True
callback = callbacks()
```

```
history = model.fit(df_train_reduced, y_train, epochs = 1000, callbacks=[callback])
```

Akurasi yang dihasilkan adalah sebesar 54%

```
Epoch 1000/1000
19/19 [==============================] - 0s 4ms/step - loss: 545833152.0000 - accuracy: 0.5461 - precision: 0.6346 - recall: 0.6804
```

Percobaan ke-2
Menggunakan nilai learning rate sebesar 0.5

```
import tensorflow as tf

model = tf.keras.Sequential (
    [
        tf.keras.layers.Dense(128, activation = 'relu'),
        tf.keras.layers.Dense(64),
        tf.keras.layers.Dense(32),
        tf.keras.layers.Dense(16),
        tf.keras.layers.Dense(4),
        tf.keras.layers.Dense(2),
        tf.keras.layers.Dense(1, activation = 'sigmoid'),
    ]
)

model.compile(loss = 'binary_crossentropy', optimizer = tf.optimizers.Adam(learning_rate=0.5), metrics = ['accuracy', 'Precision', 'Recall'])
```

```
target = 0.9
class callbacks(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs = None):
        if logs.get('accuracy') >= target:
            print('\nFor Epoch', epoch, '\nAkurasi telah mencapai = %2.2f%%' %(logs['accuracy']*100), 'proses training selesai.')
            self.model.stop_training = True
callback = callbacks()

history = model.fit(df_train_reduced, y_train, epochs = 1000, callbacks=[callback])
```

Akurasi yang dihasilkan adalah sebesar 53%

```
Epoch 1000/1000
19/19 [==============================] - 0s 5ms/step - loss: 12226.4053 - accuracy: 0.5395 - precision: 0.6406 - recall: 0.6340
```

Percobaan ke-3

Menggunakan nilai learning rate sebesar 0.05

```
import tensorflow as tf

model = tf.keras.Sequential (
    [
        tf.keras.layers.Dense(128, activation = 'relu'),
        tf.keras.layers.Dense(64),
        tf.keras.layers.Dense(32),
        tf.keras.layers.Dense(16),
        tf.keras.layers.Dense(4),
        tf.keras.layers.Dense(2),
        tf.keras.layers.Dense(1, activation = 'sigmoid'),
    ]
)

model.compile(loss = 'binary_crossentropy', optimizer = tf.optimizers.Adam(learning_rate=0.05), metrics = ['accuracy', 'Precision', 'Recall'])
```

```
target = 0.9
class callbacks(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs = None):
        if logs.get('accuracy') >= target:
            print('\nFor Epoch', epoch, '\nAkurasi telah mencapai = %2.2f%%' %(logs['accuracy']*100), 'proses training selesai.')
            self.model.stop_training = True
callback = callbacks()
```

```
history = model.fit(df_train_reduced, y_train, epochs = 1000, callbacks=[callback])
```

```
Epoch 1/1000
19/19 [==============================] - 1s 2ms/step - loss: 1.2650 - accuracy: 0.6776 - precision: 0.7791 - recall: 0.6907
Epoch 2/1000
19/19 [==============================] - 0s 3ms/step - loss: 0.2611 - accuracy: 0.8997 - precision: 0.9337 - recall: 0.9072
Epoch 3/1000
 1/19 [>.............................] - ETA: 0s - loss: 0.3611 - accuracy: 0.8438 - precision: 0.7778 - recall: 0.9333
For Epoch 2
Akurasi telah mencapai = 90.79% proses training selesai.
19/19 [==============================] - 0s 3ms/step - loss: 0.2213 - accuracy: 0.9079 - precision: 0.9346 - recall: 0.9201
```

Akurasi yang dihasilkan adalah sebesar 90%

Percobaan pada DataTest

Setelah melakukan percobaan, learning rate yang baik adalah sebesar 0.05, kita coba melakukan proses modeling pada data test dengan menerapkan learning rate sebesar 0.05.

```
import tensorflow as tf

model = tf.keras.Sequential (
    [
        tf.keras.layers.Dense(128, activation = 'relu'),
        tf.keras.layers.Dense(64),
        tf.keras.layers.Dense(32),
        tf.keras.layers.Dense(16),
        tf.keras.layers.Dense(4),
        tf.keras.layers.Dense(2),
        tf.keras.layers.Dense(1, activation = 'sigmoid'),
    ]
)

model.compile(loss = 'binary_crossentropy', optimizer = tf.optimizers.Adam(learning_rate=0.05), metrics = ['accuracy', 'Precision', 'Recall'])
```

```
target = 0.9
class callbacks(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs = None):
        if logs.get('accuracy') >= target:
            print('\nFor Epoch', epoch, '\nAkurasi telah mencapai = %2.2f%%' %(logs['accuracy']*100), 'proses training selesai.')
            self.model.stop_training = True
callback = callbacks()
```

```
history = model.fit(df_train_reduced, y_train, epochs = 1000, callbacks=[callback])
```

```
Epoch 1/1000
 1/19 [>.............................] - ETA: 0s - loss: 0.1784 - accuracy: 0.9375 - precision: 0.9167 - recall: 1.0000
For Epoch 0
Akurasi telah mencapai = 90.46% proses training selesai.
19/19 [==============================] - 0s 2ms/step - loss: 0.2197 - accuracy: 0.9046 - precision: 0.9297 - recall: 0.9201
```

```
y_train_pred = model.predict(df_train_reduced)
y_test_pred = model.predict(df_test_reduced)
```

```
19/19 [==============================] - 0s 2ms/step
5/5 [==============================] - 0s 3ms/step
```

```
y_train_pred_class = [1 if prob > 0.5 else 0 for prob in np.ravel(y_train_pred)]
y_test_pred_class = [1 if prob > 0.5 else 0 for prob in np.ravel(y_test_pred)]
```

```
from sklearn.metrics import confusion_matrix

print(confusion_matrix(y_train,y_train_pred_class))
print(confusion_matrix(y_test,y_test_pred_class))
```

```
[[200  20]
 [ 15 373]]
[[53  2]
 [ 3 94]]
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score
print(f'Accuracy : {accuracy_score(y_test, y_test_pred_class)*100:.2f}%')
print(f'Precision : {precision_score(y_test, y_test_pred_class)*100:.2f}%')
print(f'Recall : {recall_score(y_test, y_test_pred_class)*100:.2f}%')
```

```
Accuracy : 89.47%
Precision : 87.16%
Recall : 97.94%
```

Hasil akurasi pada data test jauh lebih besar dibandingkan SLP yaitu sebesar 89%

BAB 5
KESIMPULAN

Berdasarkan pengujian yang dilakukan di atas, dengan menggunakan algoritma *Artificial* Neural Network dan metode Holdout validation untuk membangun model dengan perbandingan pembagian data sebesar 80% untuk data training dan 20% untuk data uji, didapatkan hasil sebagai berikut:

1. Percobaan menggunakan algoritma *Single Layer Perceptron*

   Berdasarkan hasil percobaan menggunakan model *Single Layer Perceptron*, didapatkan hasil akurasi paling maksimal sebanyak 70% untuk data train dan 56% untuk data tes, beberapa hal yang mempengaruhi model *Single Layer Perceptron* adalah besar hyper parameter yang digunakan, yaitu learning rate yang digunakan, serta penggunaan w1,w2, dan b yang digunakan turut mempengaruhi besaran akurasi yang dihasilkan, dan juga jumlah iterasi atau pengulangan yang dilakukan akan berpengaruh pada hasil akhir pemodelan.

2. Percobaan menggunakan algoritma *Multi Layer Perceptron* menggunakan library Tensorflow

   Berdasarkan hasil percobaan menggunakan model *Multi Layer Perceptron* dengan bantuan library Tensorflow, dengan menggunakan 5 hidden layer, didapatkan hasil akurasi paling maksimal sebanyak 90% untuk data train dan 89% untuk data tes. beberapa hal yang mempengaruhi model *Multi Layer Perceptron* adalah besar hyper parameter yang digunakan yaitu learning rate yang digunakan. Jumlah hidden layer pun akan berpengaruh terhadap hasil pemodelan serta jumlah epoch atau perulangan yang dilakukan akan berpengaruh terhadap hasil akhir pemodelan.

Multi Layer Perceptron jelas memiliki tingkat akurasi yang lebih besar dibandingkan dengan model *Single Layer Perceptron* hal tersebut dikarenakan jumlah layer yang digunakan pada *Multi Layer Perceptron* lebih banyak dibandingkan model *Single Layer Perceptron*.

Lampiran :
Link colab SLP :
https://colab.research.google.com/drive/1F6_ahyE_gx3WrWJfd9g0fWv_vcGngZ4O?usp=sharing

Link colab MLP:
https://colab.research.google.com/drive/1Oy-ScIhnhx4e7lDI4lzQSj2fcq3kiUXi?usp=sharing

Link Video Presentasi:
https://drive.google.com/drive/folders/1vsuRuA4zJunDdNUa00Jyr_7urtIRonSU?usp=share_link