

Laporan Tugas Pemrograman Learning



Dosen Pembimbing:
TJOKORDA AGUNG BUDI WIRAYUDA, S.T., M.T.
CII3C3-IF-44-11

Disusun oleh:
1301204125 Ryan Chandra Hadi

S1 Informatika
Universitas Telkom
Bandung, Jawa Barat 2022

Kata Pengantar

Dengan mengucapkan puji dan rasa syukur kepada Allah SWT yang telah memberikan rahmat sehingga kita dapat menyelesaikan tugas dari mata kuliah Pembelajaran Mesin dengan tema “Implementasi AAN/MLP/RNN/LSTM/CNN” dengan benar dan tepat waktu.

Untuk memenuhi nilai tugas pada mata kuliah Pembelajaran Mesin, maka dibuatkan tugas yang dapat saya selesaikan. Tidak hanya itu, tujuan dari pembuatan laporan dan pengerjaan tugas ini adalah untuk menambah wawasan tentang pembahasan Implementasi ANN bagi kita semua.

Saya mengucapkan terima kasih kepada semua pihak mulai dari Pak Tjokorda Agung Budi Wirayuda, S.T., M.T. selaku dosen pembimbing yang telah memberikan saya tugas besar untuk membuat Algoritma ANN.

Saya sangat menyadari laporan yang saya susun masih jauh dari kata sempurna. Tetapi saya akan terus berusaha untuk selalu menjadi lebih baik untuk kedepannya.

Pernyataan : Saya mengerjakan tugas ini dengan cara yang tidak melanggar aturan perkuliahan dan kode etik akademisi.

Bandung, 7 November 2022

Ryan Chandra Hadi

BAB I

PENDAHULUAN

Case-Based 1

Scenario

Mengikuti keberhasilan tugas sebelumnya, Anda diberi kesempatan lebih lanjut untuk mengesankan atasan Anda mengenai kemampuan Anda untuk menganalisis data. Anda diminta untuk melakukan beberapa analisis dan menghasilkan seperangkat aturan yang berguna menggunakan dataset berikut:

Kumpulan data berikut tersedia online, tautan ke kumpulan data adalah sebagai berikut:

[untuk NIM akhir GENAP gunakan data ini] <https://archive.ics.uci.edu/ml/datasets/Arrhythmia>

[untuk NIM akhir GANJIL gunakan data ini] <https://archive.ics.uci.edu/ml/datasets/Audit+Data>

Anda perlu mempelajari data dengan hati-hati. Kemudian pilih teknik pra-pemrosesan data apa yang akan dilakukan untuk meningkatkan kualitas data tersebut. Akan ada banyak hal yang harus diuraikan dan kemudian Anda harus mengumpulkan case-based ini sebagai karya individu.

Hint: Anda bebas memilih satu dari tiga alat analisis data yaitu Weka, R, atau Python untuk membantu

Anda menganalisis data dan menunjukkan pra-pemrosesan data yang diperlukan.

Tugas Anda

Tujuan dari tugas ini yaitu Anda diharapkan mampu menjelaskan, mengimplementasikan, menganalisis, dan mendesain teknik pembelajaran mesin supervised learning yaitu AAN/MLP/RNN/LSTM/CNN. Pertama, selidiki masalah kualitas data yang telah diberikan di atas. Jelaskan keputusan Anda mengenai pendekatan pra-pemrosesan data. Jelajahi kumpulan data dengan meringkas data menggunakan statistik dan mengidentifikasi masalah kualitas data apa pun. Tidak ada batasan jumlah ringkasan yang akan dilaporkan tetapi Anda diharapkan hanya melaporkan yang paling relevan. Kedua, pilih salah satu dari metode unsupervised learning yang telah dipelajari yaitu AAN/MLP/RNN/LSTM/CNN. Anda hanya perlu memilih satu metode untuk diterapkan. Gunakan algoritma tersebut untuk memberikan beberapa output/outcome dengan menggunakan variasi hyperparameter, kemudian menganalisis hasilnya.

BAB II

PEMBAHASAN

2.1 Ikhtisar Kumpulan Data yang Dipilih

Pada bagian ini saya mendapatkan data ganjil dengan nim yang diakhiri dengan “5” maka dari itu saya akan memakai data audit dan trial sebagai acuan pengerjaan soal.

[3] audit

	Sector_score	LOCATION_ID	PARA_A	Score_A	Risk_A	PARA_B	Score_B	Risk_B	TOTAL	numbers	...	Risk_E	History	Prob	Risk_F	Score	Inherent_Risk	CONTROL_RISK	Detection_Risk	Audit_Risk	Risk
0	3.89	23	4.18	0.6	2.508	2.50	0.2	0.500	6.68	5.0	...	0.4	0	0.2	0.0	2.4	8.574	0.4	0.5	1.7148	1
1	3.89	6	0.00	0.2	0.000	4.83	0.2	0.966	4.83	5.0	...	0.4	0	0.2	0.0	2.0	2.554	0.4	0.5	0.5108	0
2	3.89	6	0.51	0.2	0.102	0.23	0.2	0.046	0.74	5.0	...	0.4	0	0.2	0.0	2.0	1.548	0.4	0.5	0.3096	0
3	3.89	6	0.00	0.2	0.000	10.80	0.6	6.480	10.80	6.0	...	0.4	0	0.2	0.0	4.4	17.530	0.4	0.5	3.5060	1
4	3.89	6	0.00	0.2	0.000	0.08	0.2	0.016	0.08	5.0	...	0.4	0	0.2	0.0	2.0	1.416	0.4	0.5	0.2832	0
...
771	55.57	9	0.49	0.2	0.098	0.40	0.2	0.080	0.89	5.0	...	0.4	0	0.2	0.0	2.0	1.578	0.4	0.5	0.3156	0
772	55.57	16	0.47	0.2	0.094	0.37	0.2	0.074	0.84	5.0	...	0.4	0	0.2	0.0	2.0	1.568	0.4	0.5	0.3136	0
773	55.57	14	0.24	0.2	0.048	0.04	0.2	0.008	0.28	5.0	...	0.4	0	0.2	0.0	2.0	1.456	0.4	0.5	0.2912	0
774	55.57	18	0.20	0.2	0.040	0.00	0.2	0.000	0.20	5.0	...	0.4	0	0.2	0.0	2.0	1.440	0.4	0.5	0.2880	0
775	55.57	15	0.00	0.2	0.000	0.00	0.2	0.000	0.00	5.0	...	0.4	0	0.2	0.0	2.0	1.464	0.4	0.5	0.2928	0

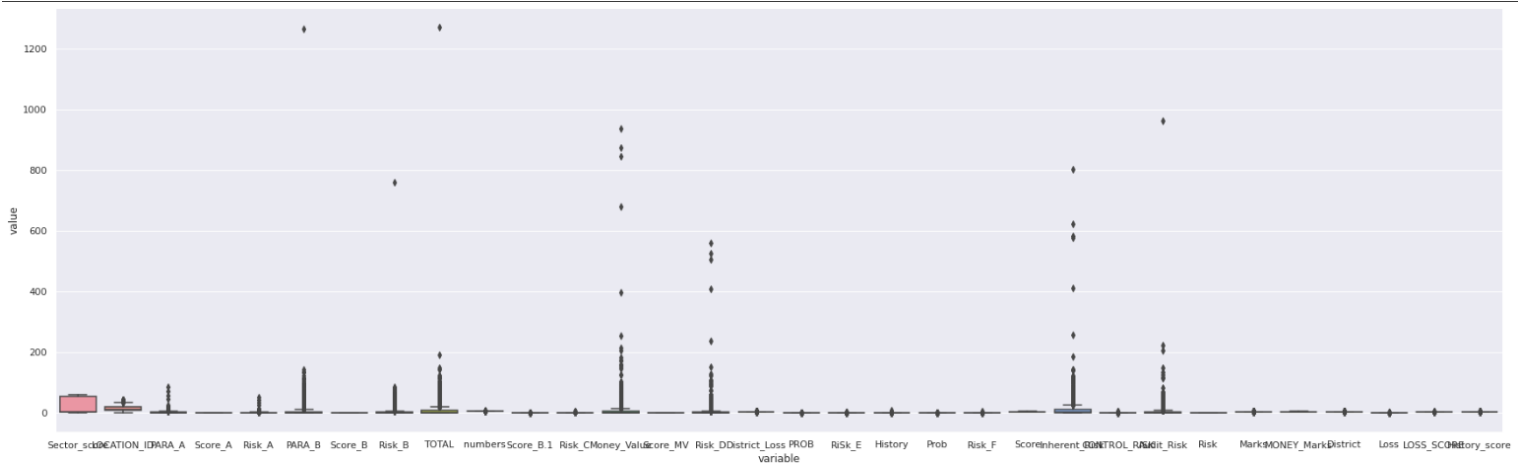
776 rows x 27 columns

Data yang bersifat supervised learning yang memiliki nilai target yang ada di kolom *Risk*, data ini merupakan data *history risk factor* dimana kita diminta untuk melakukan research mengenai dataset ini untuk bisa memperkirakan perusahaan mana saja yang bisa diklasifikasikan sebagai perusahaan yang curang berdasarkan history fakta sekarang yang ada. Data ini juga memiliki data trial sebagai dengan kolom dengan kolom yang berbeda tetapi memiliki value yang sama dengan data audit asli.

[4] trial

	Sector_score	LOCATION_ID	PARA_A	SCORE_A	PARA_B	SCORE_B	TOTAL	numbers	Marks	Money_Value	MONEY_Marks	District	Loss	LOSS_SCORE	History	History_score	Score	Risk
0	3.89	23	4.18	6	2.50	2	6.68	5.0	2	3.38	2	2	0	2	0	2	2.4	1
1	3.89	6	0.00	2	4.83	2	4.83	5.0	2	0.94	2	2	0	2	0	2	2.0	0
2	3.89	6	0.51	2	0.23	2	0.74	5.0	2	0.00	2	2	0	2	0	2	2.0	0
3	3.89	6	0.00	2	10.80	6	10.80	6.0	6	11.75	6	2	0	2	0	2	4.4	1
4	3.89	6	0.00	2	0.08	2	0.08	5.0	2	0.00	2	2	0	2	0	2	2.0	0
...
771	55.57	9	0.49	2	0.40	2	0.89	5.0	2	0.00	2	2	0	2	0	2	2.0	0
772	55.57	16	0.47	2	0.37	2	0.84	5.0	2	0.00	2	2	0	2	0	2	2.0	0
773	55.57	14	0.24	2	0.04	2	0.28	5.0	2	0.00	2	2	0	2	0	2	2.0	0
774	55.57	18	0.20	2	0.00	2	0.20	5.0	2	0.00	2	2	0	2	0	2	2.0	0
775	55.57	15	0.00	2	0.00	2	0.00	5.0	2	0.32	2	2	0	2	0	2	2.0	0

776 rows x 18 columns



Ini merupakan persebaran data pada data audit, terlihat di sini masih banyak sekali data yang memiliki outliers.

Setelah itu kita akan menggabungkan data audit dengan data trial dengan merekonstruksi nama kolom agar bisa sesuai dengan data yang ada di data audit. Setelah itu penyesuaian juga dilakukan dengan membagi nilai pada kolom *Score_A* yang ada pada data trial kita bagi dengan 10 agar datanya selaras dengan data audit sebelum gabungkan data tersebut.

```
[10] trial['Score_A'] = trial['Score_A']/10
     trial['Score_B'] = trial['Score_B']/10
```

```
] test = np.intersect1d(audit.columns, trial.columns)
test

array(['History', 'LOCATION_ID', 'Money_Value', 'PARA_A', 'PARA_B',
       'Score', 'Score_A', 'Score_B', 'Sector_score', 'TOTAL', 'numbers'],
      dtype=object)
```

```
merged = pd.merge(audit, trial, how='outer', on = ['History', 'LOCATION_ID', 'Money_Value', 'PARA_A', 'PARA_B',
                                                  'Score', 'Score_A', 'Score_B', 'Sector_score', 'TOTAL', 'numbers'])
merged.columns

Index(['Sector_score', 'LOCATION_ID', 'PARA_A', 'Score_A', 'Risk_A', 'PARA_B',
       'Score_B', 'Risk_B', 'TOTAL', 'numbers', 'Score_B.1', 'Risk_C',
       'Money_Value', 'Score_MV', 'Risk_D', 'District_Loss', 'PROB', 'Risk_E',
       'History', 'Prob', 'Risk_F', 'Score', 'Inherent_Risk', 'CONTROL_RISK',
       'Detection_Risk', 'Audit_Risk', 'Risk', 'Marks', 'MONEY_Marks',
       'District', 'Loss', 'LOSS_SCORE', 'History_score', 'Risk_trial'],
      dtype='object')
```

Poin kunci terkait informasi mengenai pendekatan pra-pemrosesan ini ada pada merge data audit dengan trial dengan beberapa ketentuan yang sudah disesuaikan agar datanya bisa fit dan sesuai dengan data yang ingin di gabung. Sehingga menurut saya kualitas yang ada pada data yang sudah digabung menjadi lebih baik dibandingkan sebelumnya. Tentunya dengan beberapa penyesuaian seperti pengurangan dimensi dll.

2.2 Ringkasan pra-pemrosesan data yang diusulkan

Pada bagian pra-pemrosesan data ini saya memakai beberapa metode atau teknik yang sekiranya saya harapkan bisa membuat data tersebut menjadi lebih baik lagi sebelum dijadikan model untuk kedepannya. Pada kesempatan ini saya menggunakan metode **reduksi dimensi** dengan tujuan untuk mengurangi **penggunaan memori dan waktu** dan juga membantu mengeliminasi data yang **tidak relevan atau noise**.

```
for column_name in df:  
    print(df[column_name].value_counts())
```

Di atas ini saya akan meng-outputkan semua nilai kolom pada data df tersebut,

```
Name: CONTROL_RISK, dtype: int64  
0.5      810  
Name: Detection_Risk, dtype: int64  
0.5      810
```

Setelah itu kita selidiki data yang hanya memiliki satu unique data, yaitu ada pada kolom *Detection_Risk* yang hanya memiliki satu nilai yaitu 0.5, karena ini saya anggap hanya akan membebankan komputasi. Maka dari itu kita akan menghapus data kolom tersebut dengan code seperti berikut.

```
df = df.drop(['Detection_Risk'], axis=1)  
df.info()
```

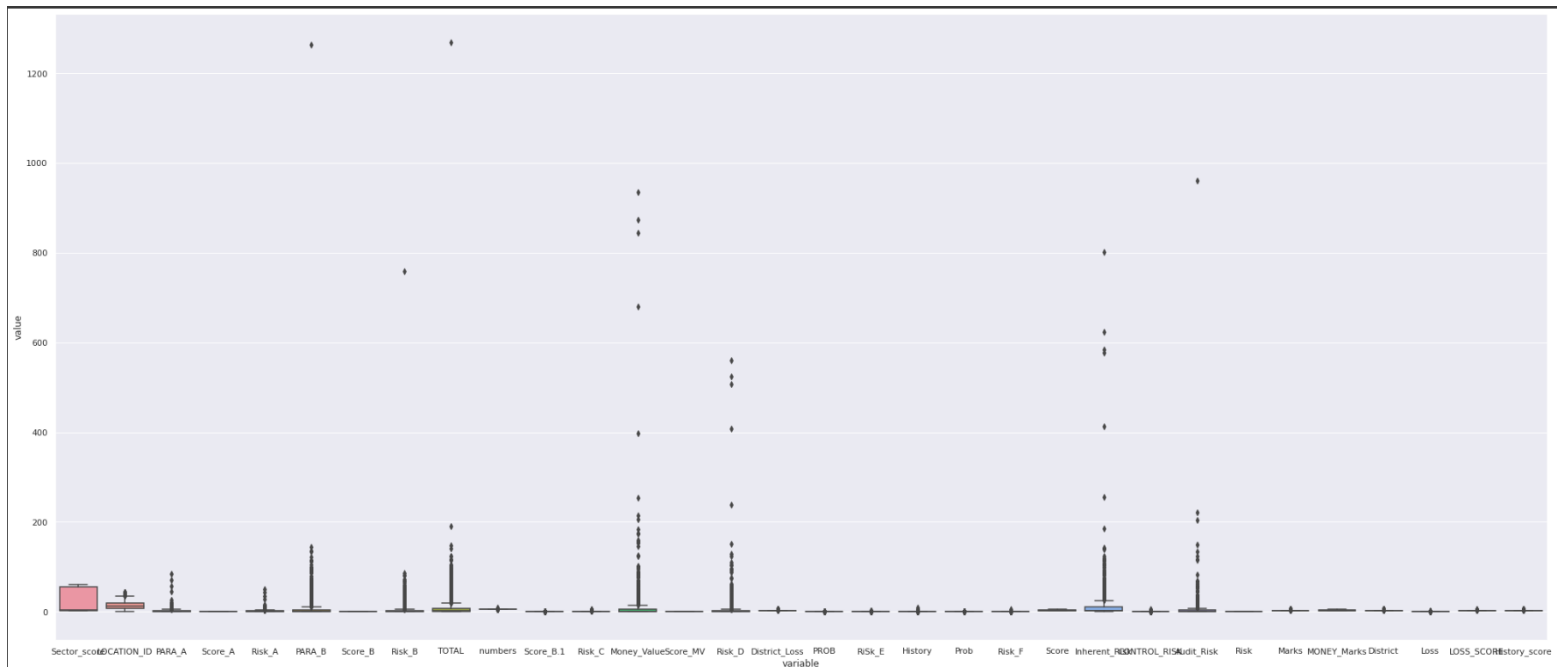
Setelah kita perhatikan juga, ada beberapa data yang berbeda dari value nilai yang seharusnya ada seperti berikut.

```
[ ] df['LOCATION_ID'].sort_values()  
  
248      1  
247      1  
242      1  
488      1  
489      1  
...  
200      9  
54       9  
353      LOHARU  
357      NUH  
369      SAFIDON  
Name: LOCATION_ID, Length: 810, dtype: object  
  
[ ] df=df.drop(353)  
df=df.drop(357)  
df=df.drop(369)
```

Pada kolom *LOCATION_ID* hampir 98% data berisi objek berupa angka, tetapi setelah diperhatikan ada tiga data yang memiliki nilai seperti berikut,

Loharu, Nuh, Safidon

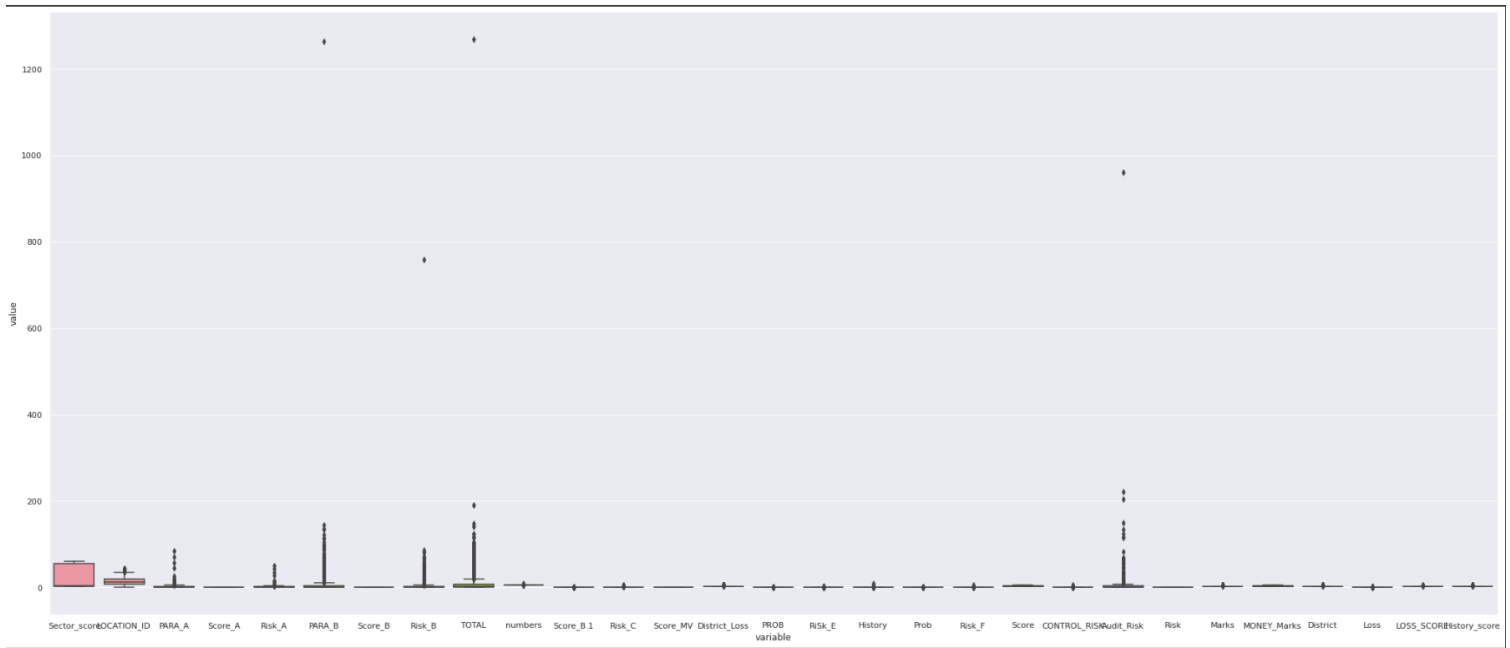
Tiga nilai tersebut akan di drop untuk mempermudah kita dalam mengolah data sebelum kita buat modelnya.



Dalam boxplot di atas juga ada beberapa data yang memiliki pencilan atau outliers yang sangat banyak dalam hal ini, saya memutuskan untuk men-drop kolom yang memiliki data dengan pencilan yang cukup mendominasi yaitu sebagai berikut,

```
df = df.drop(['Money_Value'], axis = 1)
df = df.drop(['Risk_D'], axis = 1)
df = df.drop(['Inherent_Risk'], axis = 1)
```


Dan akan jadi seperti berikut,



Terlihat bahwa sudah tidak ada kolom *Money_value*, *inherent_risk*, and *risk_D*.

```
[79] class_df = df.drop("Audit_Risk", axis = 1)
```

```
[80] klasifikasi_X = class_df.drop(["Risk"], axis = 1)
      klasifikasi_y = class_df["Risk"]
```

```
[81] X_train_awal, X_test_awal, y_train, y_test = train_test_split(klasifikasi_X, klasifikasi_y, test_size = 0.25, random_state = 1)
```

Selanjutnya kita akan membelah data menjadi dua yaitu klasifikasi x dan klasifikasi y sebelum kita masuk ke dalam pembuatan model ANN sebagai berikut, dimana klasifikasi x berisikan semua data pada variabel *class_df* kecuali kolom *risk* dikarenakan kolom tersebut akan dijadikan acuan sebagai evaluasi nanti, sedangkan klasifikasi y hanya berisikan data kolom *risk* untuk menjadi bahan evaluasi setelah klasifikasi x melakukan training dan learning.

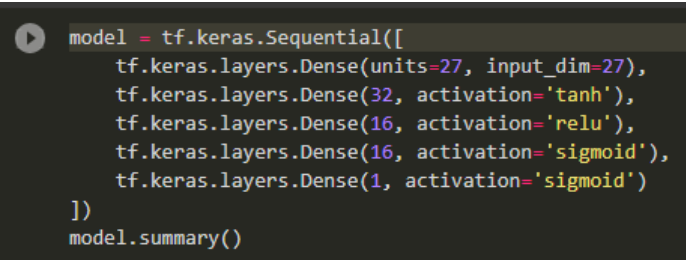
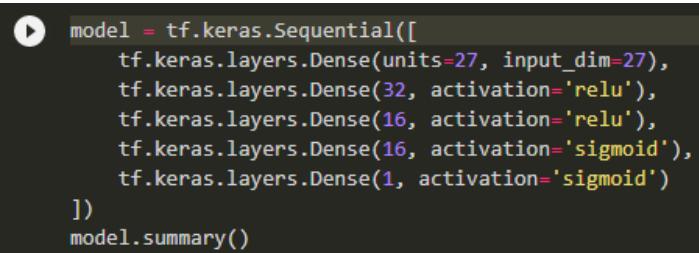
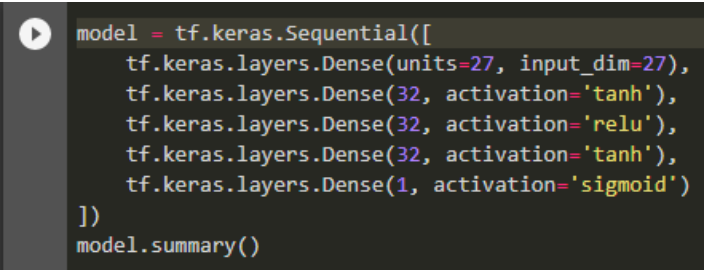
```
[82] scale = MinMaxScaler()  
      X_train = scale.fit_transform(X_train_awal)  
      X_test = scale.transform(X_test_awal)
```

Selanjutnya setelah kita split menjadi dua (klasifikasi X dan Y) kita melakukan scaling agar data bisa terkontrol (data berisikan value dari 0..1)

2.3 Menerapkan Algoritma



Pada kesempatan ini saya menggunakan metode ANN (Artificial Neural Network) untuk mengerjakan tugas ini. Metode Artificial Neural Network (ANN) sendiri merupakan suatu pendekatan model kecerdasan yang di dasari akan cara bekerja struktur otak manusia dan kemudian diimplementasikan menggunakan program komputer yang mampu menyelesaikan sejumlah proses perhitungan selama proses learning berlangsung.

Dengan menggunakan ini diharapkan mendapatkan nilai training dan nilai validasi yang sesuai dengan harapan (0.8.. ++) dan tidak terjadi overfitting model atau bad model.

Metode yang dicoba	Nilai Epoch Terakhir (e=100)
 <pre>model = tf.keras.Sequential([tf.keras.layers.Dense(27, input_dim=27), tf.keras.layers.Dense(32, activation='tanh'), tf.keras.layers.Dense(16, activation='relu'), tf.keras.layers.Dense(16, activation='sigmoid'), tf.keras.layers.Dense(1, activation='sigmoid')]) model.summary()</pre>	0.9912
 <pre>model = tf.keras.Sequential([tf.keras.layers.Dense(27, input_dim=27), tf.keras.layers.Dense(32, activation='relu'), tf.keras.layers.Dense(16, activation='relu'), tf.keras.layers.Dense(16, activation='sigmoid'), tf.keras.layers.Dense(1, activation='sigmoid')]) model.summary()</pre>	0.9912
 <pre>model = tf.keras.Sequential([tf.keras.layers.Dense(27, input_dim=27), tf.keras.layers.Dense(32, activation='tanh'), tf.keras.layers.Dense(32, activation='relu'), tf.keras.layers.Dense(32, activation='tanh'), tf.keras.layers.Dense(1, activation='sigmoid')]) model.summary()</pre>	0.9947

<pre>[44] model = tf.keras.Sequential([tf.keras.layers.Dense(units=27, input_dim=27), tf.keras.layers.Dense(32, activation='tanh'), tf.keras.layers.Dense(32, activation='relu'), tf.keras.layers.Dense(1, activation='sigmoid')]) model.summary()</pre>	0.9877
<pre>▶ model = tf.keras.Sequential([tf.keras.layers.Dense(units=27, input_dim=27), tf.keras.layers.Dense(32, activation='tanh'), tf.keras.layers.Dense(32, activation='relu'), tf.keras.layers.Dense(1, activation='tanh')]) model.summary()</pre>	0.9596

Metode yang dicoba	Nilai Epoch Terakhir (e=50)
<pre>▶ model = tf.keras.Sequential([tf.keras.layers.Dense(units=27, input_dim=27), tf.keras.layers.Dense(32, activation='tanh'), tf.keras.layers.Dense(16, activation='relu'), tf.keras.layers.Dense(16, activation='sigmoid'), tf.keras.layers.Dense(1, activation='sigmoid')]) model.summary()</pre>	0.9825
<pre>▶ model = tf.keras.Sequential([tf.keras.layers.Dense(units=27, input_dim=27), tf.keras.layers.Dense(32, activation='relu'), tf.keras.layers.Dense(16, activation='relu'), tf.keras.layers.Dense(16, activation='sigmoid'), tf.keras.layers.Dense(1, activation='sigmoid')]) model.summary()</pre>	0.9807

 <pre>model = tf.keras.Sequential([tf.keras.layers.Dense(units=27, input_dim=27), tf.keras.layers.Dense(32, activation='tanh'), tf.keras.layers.Dense(32, activation='relu'), tf.keras.layers.Dense(32, activation='tanh'), tf.keras.layers.Dense(1, activation='sigmoid')]) model.summary()</pre>	0.9825
<pre>[44] model = tf.keras.Sequential([tf.keras.layers.Dense(units=27, input_dim=27), tf.keras.layers.Dense(32, activation='tanh'), tf.keras.layers.Dense(32, activation='relu'), tf.keras.layers.Dense(1, activation='sigmoid')]) model.summary()</pre>	0.9842
 <pre>model = tf.keras.Sequential([tf.keras.layers.Dense(units=27, input_dim=27), tf.keras.layers.Dense(32, activation='tanh'), tf.keras.layers.Dense(32, activation='relu'), tf.keras.layers.Dense(1, activation='tanh')]) model.summary()</pre>	0.9772

Berikut merupakan link colabnya:




<https://colab.research.google.com/drive/1qBWuf06F5BbkZVHk5T1P5synjlsDkEIG?usp=sharing>

Berikut merupakan link youtube:

<https://youtu.be/hd10o5bFguU>

2.4 Evaluasi Hasil

Random State = 1





Metode yang dicoba	Nilai Epoch Terakhir (e=100)	Nilai Evaluasi
 <pre>model = tf.keras.Sequential([tf.keras.layers.Dense(units=27, input_dim=27), tf.keras.layers.Dense(32, activation='tanh'), tf.keras.layers.Dense(16, activation='relu'), tf.keras.layers.Dense(16, activation='sigmoid'), tf.keras.layers.Dense(1, activation='sigmoid')]) model.summary()</pre>	0.9912	0.9947
 <pre>model = tf.keras.Sequential([tf.keras.layers.Dense(units=27, input_dim=27), tf.keras.layers.Dense(32, activation='relu'), tf.keras.layers.Dense(16, activation='relu'), tf.keras.layers.Dense(16, activation='sigmoid'), tf.keras.layers.Dense(1, activation='sigmoid')]) model.summary()</pre>	0.9912	0.9947
 <pre>model = tf.keras.Sequential([tf.keras.layers.Dense(units=27, input_dim=27), tf.keras.layers.Dense(32, activation='tanh'), tf.keras.layers.Dense(32, activation='relu'), tf.keras.layers.Dense(32, activation='tanh'), tf.keras.layers.Dense(1, activation='sigmoid')]) model.summary()</pre>	0.9947	0.9947

<pre>[44] model = tf.keras.Sequential([tf.keras.layers.Dense(units=27, input_dim=27), tf.keras.layers.Dense(32, activation='tanh'), tf.keras.layers.Dense(32, activation='relu'), tf.keras.layers.Dense(1, activation='sigmoid')]) model.summary()</pre>	0.9877	0.9947
<pre>▶ model = tf.keras.Sequential([tf.keras.layers.Dense(units=27, input_dim=27), tf.keras.layers.Dense(32, activation='tanh'), tf.keras.layers.Dense(32, activation='relu'), tf.keras.layers.Dense(1, activation='tanh')]) model.summary()</pre>	0.9596	0.9947

Metode yang dicoba	Nilai Epoch Terakhir (e=50)	Nilai Evaluasi
<pre>▶ model = tf.keras.Sequential([tf.keras.layers.Dense(units=27, input_dim=27), tf.keras.layers.Dense(32, activation='tanh'), tf.keras.layers.Dense(16, activation='relu'), tf.keras.layers.Dense(16, activation='sigmoid'), tf.keras.layers.Dense(1, activation='sigmoid')]) model.summary()</pre>	0.9825	0.9895
<pre>▶ model = tf.keras.Sequential([tf.keras.layers.Dense(units=27, input_dim=27), tf.keras.layers.Dense(32, activation='relu'), tf.keras.layers.Dense(16, activation='relu'), tf.keras.layers.Dense(16, activation='sigmoid'), tf.keras.layers.Dense(1, activation='sigmoid')]) model.summary()</pre>	0.9807	0.9842

 <pre> model = tf.keras.Sequential([tf.keras.layers.Dense(units=27, input_dim=27), tf.keras.layers.Dense(32, activation='tanh'), tf.keras.layers.Dense(32, activation='relu'), tf.keras.layers.Dense(32, activation='tanh'), tf.keras.layers.Dense(1, activation='sigmoid')]) model.summary() </pre>	0.9825	0.9947
<pre> [44] model = tf.keras.Sequential([tf.keras.layers.Dense(units=27, input_dim=27), tf.keras.layers.Dense(32, activation='tanh'), tf.keras.layers.Dense(32, activation='relu'), tf.keras.layers.Dense(1, activation='sigmoid')]) model.summary() </pre>	0.9842	0.9947
 <pre> model = tf.keras.Sequential([tf.keras.layers.Dense(units=27, input_dim=27), tf.keras.layers.Dense(32, activation='tanh'), tf.keras.layers.Dense(32, activation='relu'), tf.keras.layers.Dense(1, activation='tanh')]) model.summary() </pre>	0.9772	0.9895

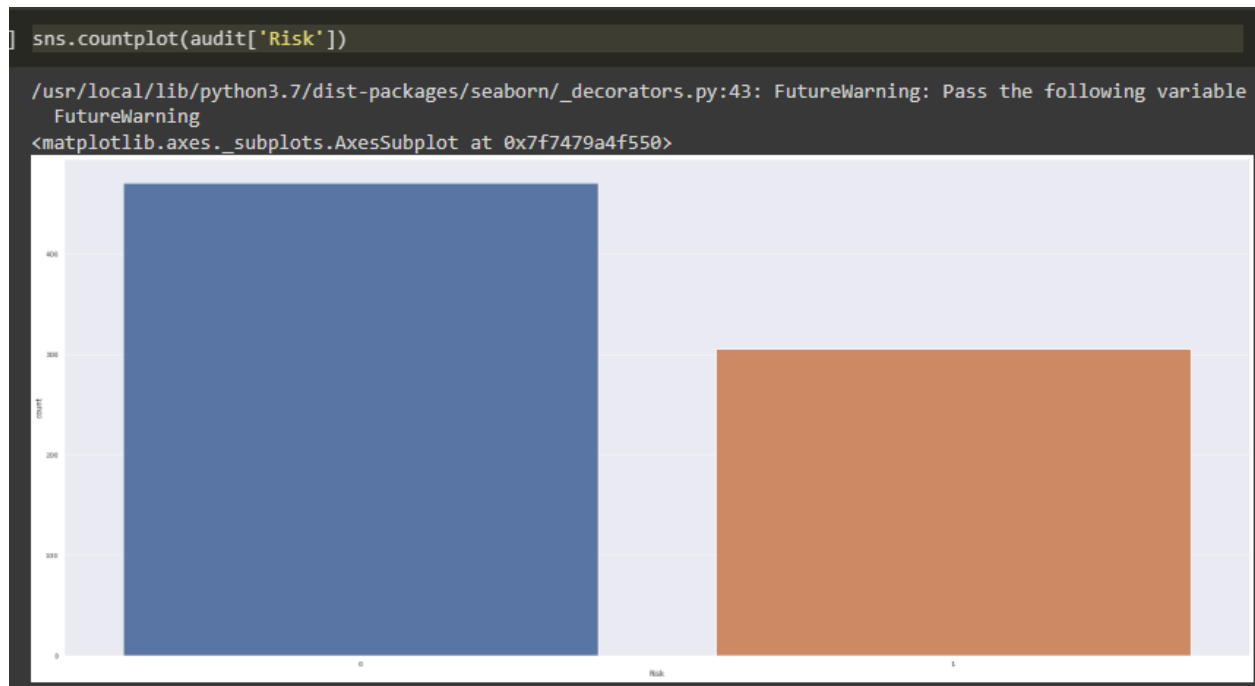
Random State = 5

Metode yang dicoba	Nilai Epoch Terakhir (e=100)	Nilai Evaluasi
 <pre>model = tf.keras.Sequential([tf.keras.layers.Dense(units=27, input_dim=27), tf.keras.layers.Dense(32, activation='tanh'), tf.keras.layers.Dense(16, activation='relu'), tf.keras.layers.Dense(16, activation='sigmoid'), tf.keras.layers.Dense(1, activation='sigmoid')]) model.summary()</pre>	0.9842	0.9895
 <pre>model = tf.keras.Sequential([tf.keras.layers.Dense(units=27, input_dim=27), tf.keras.layers.Dense(32, activation='relu'), tf.keras.layers.Dense(16, activation='relu'), tf.keras.layers.Dense(16, activation='sigmoid'), tf.keras.layers.Dense(1, activation='sigmoid')]) model.summary()</pre>	0.9877	0.9737
 <pre>model = tf.keras.Sequential([tf.keras.layers.Dense(units=27, input_dim=27), tf.keras.layers.Dense(32, activation='tanh'), tf.keras.layers.Dense(32, activation='relu'), tf.keras.layers.Dense(32, activation='tanh'), tf.keras.layers.Dense(1, activation='sigmoid')]) model.summary()</pre>	0.9807	0.9895
<pre>[44] model = tf.keras.Sequential([tf.keras.layers.Dense(units=27, input_dim=27), tf.keras.layers.Dense(32, activation='tanh'), tf.keras.layers.Dense(32, activation='relu'), tf.keras.layers.Dense(1, activation='sigmoid')]) model.summary()</pre>	0.9789	0.9895
 <pre>model = tf.keras.Sequential([tf.keras.layers.Dense(units=27, input_dim=27), tf.keras.layers.Dense(32, activation='tanh'), tf.keras.layers.Dense(32, activation='relu'), tf.keras.layers.Dense(1, activation='tanh')]) model.summary()</pre>	0.9684	0.9842

Metode yang dicoba	Nilai Epoch Terakhir (e=50)	Nilai Evaluasi
<pre> ▶ model = tf.keras.Sequential([tf.keras.layers.Dense(units=27, input_dim=27), tf.keras.layers.Dense(32, activation='tanh'), tf.keras.layers.Dense(16, activation='relu'), tf.keras.layers.Dense(16, activation='sigmoid'), tf.keras.layers.Dense(1, activation='sigmoid')]) model.summary()</pre>	0.9789	0.9895
<pre> ▶ model = tf.keras.Sequential([tf.keras.layers.Dense(units=27, input_dim=27), tf.keras.layers.Dense(32, activation='relu'), tf.keras.layers.Dense(16, activation='relu'), tf.keras.layers.Dense(16, activation='sigmoid'), tf.keras.layers.Dense(1, activation='sigmoid')]) model.summary()</pre>	0.9772	0.9895
<pre> ▶ model = tf.keras.Sequential([tf.keras.layers.Dense(units=27, input_dim=27), tf.keras.layers.Dense(32, activation='tanh'), tf.keras.layers.Dense(32, activation='relu'), tf.keras.layers.Dense(32, activation='tanh'), tf.keras.layers.Dense(1, activation='sigmoid')]) model.summary()</pre>	0.9789	0.9842
<pre> [44] model = tf.keras.Sequential([tf.keras.layers.Dense(units=27, input_dim=27), tf.keras.layers.Dense(32, activation='tanh'), tf.keras.layers.Dense(32, activation='relu'), tf.keras.layers.Dense(1, activation='sigmoid')]) model.summary()</pre>	0.9719	0.9895
<pre> ▶ model = tf.keras.Sequential([tf.keras.layers.Dense(units=27, input_dim=27), tf.keras.layers.Dense(32, activation='tanh'), tf.keras.layers.Dense(32, activation='relu'), tf.keras.layers.Dense(1, activation='tanh')]) model.summary()</pre>	0.9667	0.9789

Dalam percobaan ini saya menggunakan dua *epoch* sebagai acuan, yang pertama e senilai 50 dan yang kedua senilai 100. Setelah itu saya juga menggunakan 2 nilai *random state* yaitu sebesar 1 dan 5, dan saya juga meng-variasikan jenis aktivasi pada setiap layer yang ada untuk mencari perbedaan atau setidaknya mendapat nilai evaluasi yang terbaik.

Dalam hal ini terlihat data yang ditampilkan cukup menarik, kenapa? Karena semua metode dari mulai penggunaan *epoch* dan *random state* yang berbeda sampai nilai aktivasi yang sudah dibuat berbeda sampai ada 5 contoh aktivasi berbeda dan ada yang dikurangi satu hidden layer pada gambar di tabel di atas. Tetapi memiliki hasil valuasinya cenderung sama yaitu semua hasilnya > 0.9600 . Pada data ini juga saya anggap tidak mengalami underfitting atau overfitting



Pengamatan terakhir saya mencoba melihat persebaran data dari nilai kolom *risk* dan hasilnya sebagai gambar di atas, dan saya menyimpulkan bahwa

- Kolom *risk* memiliki rasio yang normal (tidak berat sebelah terlalu parah).
- Data yang diolah pada bagian pre-processing sudah baik.

