UNIVERSITAS Telkom	**Assessment CLO-1: ANN (Artificial Neural Network)** **Semester Ganjil TA. 2023/2024** **Mata Kuliah Pembelajaran Mesin** **Lama Pengerjaan +/- 5 hari** **Dosen SSD – Siti Saadah**

= Ujian bersifat BUKA SEMUA REFERENSI; dengan pengerjaan di ketik langsung pada lembar ujian ini=
= Dilarang keras bekerja sama dan melakukan perbuatan curang/mencontek. Jika dilakukan, maka dianggap pelanggaran=

Kerjakan soal langsung pada kertas ujian ini.

NIM:	**Nama Mahasiswa:**	**Kelas:**	**Nilai: <diisi oleh dosen>**
1304211035 **…….**	**Khalifardy Miqdarsah** **…….**	**…..…….**	…….

Salinlah pernyataan berikut: ***Saya tidak akan melakukan kecurangan dan melanggar tata tertib dalam ujian ini. Jika saya melakukan pelanggaran, maka saya bersedia diberi sanksi nilai E untuk Mata Kuliah ini.*** *Saya tidak akan melakukan kecurangan dan melanggar tata tertib dalam ujian ini. Jika saya melakukan pelanggaran, maka saya bersedia diberi sanksi nilai E untuk Mata Kuliah ini.*	**Tanda Tangan Mahasiswa:** **…….**

Perhatikan link google colab berikut:
https://colab.research.google.com/drive/1PE9fy-ialLL3SBpx0DvoiUrnzZaQXimu?usp=sharing#scrollTo=uTJ4IYcwtFz2

Pertanyaan:

1. Sebutkan apa Input dan Output dari setiap program tersebut! Terdapat berapa studi kasus?
2. Jelaskan setiap dataset yang digunakan pada setiap studi kasus pada program tersebut!
3. Jelaskan maksud dari setiap perbaris atau per cell code program dengan detil!
4. Berikan analisis dari hasil evaluasi error atau loss yang diberikan!

Note:
Jelaskan jawaban Anda sedetail dan selengkap mungkin.
Simpan jawaban dengan menggunakan format <NIM>_<NAMA>_AssessmentCLO1

1. Terdapat 2 studi kasus yaitu studi kasus prediksi kanker payudara dan Memprediksi tulisan taangan. Pada prediksi kanker payudara inputan berupa parameter-parameter yang berkaitan dengan diagnosis kanker payudara dengan outputan berupa prediksi dengan parameter-prameter tersebut seseorang terjangkit kanker payudara. Untuk studi kasus kedua inputan berupa kumpulan bermacam-macam pola gambar tulisan tangan berbagai angka. Dengan outputan berupa jika diberikan suatu tulisan angka apakah program bisa memprediksi dengan tepat maksud tulisan tersebut.

2. Ada dua data set yang dipakai yaitu dataset breast_cancer dan dataset MNIST, dataset breast yang mempunyai dimensi 569 rows x 30 columns + 1 kolom data target, tidak mempunyai missing value, kebanyakan tipe data pada fitur numerik (float/integer), pada kolom worst_area dan mean_area memiliki outliers cukup banyak. Pada dataset MNIST memiliki size 1000 x 28 x 28 , artinya ada 1000 gambar dengan ukuran resolusi gambar 28x28 pixel dan juga untuk size dari target 1000 x 10 artinya ada 1000 record namun klasifikasinya ada 10.

Dataset breast cancer berisi data-data dengan parameter-parameter untuk keperluan diagnosis kanker payudara sedangkan Dataset MNIST berisi kumpulan gambar tulisan tangan yang berisi 10 jenis tulisan.

3.

```
import numpy as np
import pandas as pd

import tensorflow as tf
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

pada cell ini berisi library-library yang diperlukan nantinya untuk membangun model, library **numpy** suatu library yang berfungsi biasanya untuk menghandle aljabar linear seperti vektor atau matriks. **Pandas** suatu library di python yang digunakan untuk analisis dan manipulasi data. **tensorflow** adalah suatu library python yang dikembangkan oleh google untuk keperluan machine learning dan deeplearning. **RMSprop** adalah fungsi di dalam tensor flow yang akan mengembalikan nilai Root Mean Square propagation. **Tokenizer** suatu fungsi di dalam tensorflow yang berguna untuk mengkonversi kata-kata ke dalam bentuk token-token. **pad_sequence** fungsi di dalam tensor flow yang berfungsi untuk membuat panjang sekuensial suatu data tex menjadi panjang yang seragam .

```
lass FCLayer:
def __init__(self, input_size, output_size):
self.input_size = input_size
self.output_size = output_size
self.weights = np.random.randn(input_size, output_size) / np.sqrt(input_size + output_size)
self.bias = np.random.randn(1, output_size) / np.sqrt(input_size + output_size)

def forward(self, input):
self.input = input
# print(input.shape, self.weights.shape)
return np.dot(input, self.weights) + self.bias
```

```
def backward(self, output_error, learning_rate):
input_error = np.dot(output_error, self.weights.T)
weights_error = np.dot(input_error.T, output_error)
# bias_error = output_error
```

```
self.weights -= learning_rate * weights_error
self.bias -= learning_rate * output_error
return input_error
```

Pada cell ini dibuat suatu objek class **FCLayer** yang memiliki atribut berupa **input_size, output_size , weights dan bias.** nilai weights dan bias akan dibangkitkan secara random berdasarkan inputsize dan output_size nya , untuk bias bilangan random akan dibangkiatkan diantara 1 s/d output size.
class ini juga memiliki method **forward** dengan parameter **input** dan method **backward** dengan parameter output_error dan learning_rate. method forward akan mengembalikan nilai hasil dari perkalian dot vektor input dan vektor berat ditambahkan biasnya hal ini terkait dengan rumus persamaan linear y = wx + b. method backward akan mengembalikan nilai error dari hasil perkalian titik antara output_error dan matrix transpose dari weights, lalu kemudian weights error akan dihitung dari perkalian titik matriks transpose input error

dengan output error , lalu akan dihasilkan weigths terbaru dari pengurangan dengan hasil kali learning rate dan weights error dan bias yang baru hasil dari pengurangan dengan hasil kali learing_rate dengan outputs_error.

```
class ActivationLayer:
def __init__(self, activation, activation_prime):
self.activation = activation
self.activation_prime = activation_prime

def forward(self, input):
self.input = input
return self.activation(input)
```

```
def backward(self, output_error, learning_rate):
return output_error * self.activation_prime(self.input)
```

Pada cell ini terdapat class **Activation layer** yang memiliki atribut fungsi activation dan activation_prime memiliki dua method **forward** dan **backward ,** forward memiliki parameter input dan mengembalikan nilai dari fungsi activation yang di pilih. sedangkan backward memiliki paramter input berupa output_error dan learning rate dang mengembalikan nilai perkalian antara output_error dan activation_prime yang dipilih.

```
class FlattenLayer:
def __init__(self, input_shape):
self.input_shape = input_shape

def forward(self, input):
return np.reshape(input, (1, -1))
```

```
def backward(self, output_error, learning_rate):
return np.reshape(output_error, self.input_shape)
```

Pada cell ini class **FlattenLayer** memiliki atribut input_shape dengan method **forward dan backward**. forward akan mengembalikan bentuk ke adalam array satu dimensi dengan jumlah kolom yang sesuai, backward akan mengembalikan bentuk arrat sesuai dengan input_shape.

```
class SoftmaxLayer:
def __init__(self, input_size):
self.input_size = input_size

def forward(self, input):
self.input = input
tmp = np.exp(input)
self.output = tmp / np.sum(tmp)
return self.output
```

```
def backward(self, output_error, learning_rate):
input_error = np.zeros(output_error.shape)
out = np.tile(self.output.T, self.input_size)
# print(output_error.shape, out.shape)
return self.output * np.dot(output_error, np.identity(self.input_size) - out)
```

Pada cell ini class **SoftmaxLayer** memiliki atribut input_size dan mempunyai method **forward dan backward.** Method forward akan mengembalikan molao output hasil dari

pembagian eksponsial dipankatkan dengan input dibagi penjumlahah semua elemen di dalam input yang sudah dieksponesialkan.
backward mengembalikan nilai perkalian output dengan hasil dot product dari output_error , matriks identitas dikuraingi dengan out

```
def sigmoid(x):
return 1 / (1 + np.exp(-x))

def sigmoid_prime(x):
return np.exp(-x) / (1 + np.exp(-x))**2
```

```
def tanh(x):
return np.tanh(x)
```

```
def tanh_prime(x):
return 1 - np.tanh(x)**2
```

```
def relu(x):
return np.maximum(x, 0)
```

```
def relu_prime(x):
return np.array(x >= 0).astype('int')
```

Pada cell diatas terdapat 6 fungsi yang setiap fungsi memiliki input x yang berupa bilangan real. **fungsi sigmoid** mengembalikan nilai dari $1/(1 + e^{-x})$, **fungsi sigmoid_prime** mengembalikan nilai dari $e^{-x} / (1 + e^{-x})^2$. **Fungsi tanh** mengembalikan nilai dari **tanh x** . **Fungsi tanh_prime** mengembalikan nilai dari **1 - (tanh x) ^{2}. Fungsi relu** akan mengembalikan nilai maksimum diantara nol atau inputan. Fungsi **relu_prime** mengembalikan array untuk x lebih dari sama dengan nol dan tipe datanya integer.

```
def mse(y_true, y_pred):
return np.mean(np.power(y_true - y_pred, 2))

def mse_prime(y_true, y_pred):
return 2 * (y_pred - y_true) / y_pred.size
```

```
def sse(y_true, y_pred):
return 0.5 * np.sum(np.power(y_true - y_pred, 2))
```

```
def sse_prime(y_true, y_pred):
return y_pred - y_true
```

pada cell diatas terdapat 4 fungsi yang setiap fungsi memiliki 2 input yaitu y_true dan y_pred. **Fungsi mse** akan mengembalikan nilai mean square error. **Fungsi mse_prime** akan mengembalikan nilai turunan dari MSE terhadap y_pred. **Fungsi sse** akan mengembalikan nilai Sum squared error. **Fungsi SSE** akan mengembalikan nilai turunan SSE terhadap y_pred.

```
from sklearn.datasets import load_breast_cancer
data = load_breast_cancer()
data.keys()
```

Pada cell diatas mengambil data set **breast_cancer** dari library scikit-learn lalu set data tersebut disimpan di dalam variabel data, lalu **data.keys()** berfungsi untuk mendapatkan keys pada dicitionary di dalam dataset.

```
df = pd.DataFrame(data.data, columns=data.feature_names)
```

data set pada variabel data di konversi kedalam dataframe pandas dengan kolom sesuai dengan featurnya. dataframe di simpan kedalam variabel df

```
df['Target'] = data.target
```

lalu dataframe ditambahkan satu kolom bernama target dengan value data.target

```
df.isna().sum().sum()
```

baris ini berfungsi untuk menjumlahkan seluruh misiing value di dalam dataframe

```
df.info()
```

baris ini berfungsi untuk melihat informasi nama kolom berserta jumlah data yang tidak kosong dan tipe datanya.

```
df.describe()
```

baris ini untuk menampilkan informasi statistik dari data, seperti mean, standard deviasi, persentil dst.

```
import seaborn as sns
import matplotlib.pyplot as plt
def boxplot(df):
sns.set(rc={'figure.figsize':(30,9)})
sns.boxplot(x="variable", y="value", data=pd.melt(df[df.columns]))

plt.show()
```

pada cell ini dibuat suatu prosedur dengan inputan dataframe dan akan memprovide atau menampilkan suatu boxplot . Library yang dipakai adalah seaborn dan matplotlib.

```
boxplot(df)
```

fungsi boxplot dipanggil untuk menampilkan boxplot dari dataframe df.

```
from scipy import stats
z = np.abs(stats.zscore(df[df.columns[:-1]]))
```

Baris code ini menghitung nilai z-score pada setiap kolom kecuali kolom terakhir lalu diambil nilai absolute atau non negatif dari nilai zscore tersebut.

```
df1 = df[(z < 5).all(axis=1)]
```

baris ini mengambil record atau baris ketika nilai z < 5

```
sns.set(rc={'figure.figsize':(30,9)})
sns.boxplot(x="variable", y="value", data=pd.melt(df1[df1.columns]))

plt.show()
```

baris kode ini adalah untuk menampilkan boxplot dari dataframe df 1

```
df2 = df.drop(['worst area', 'mean area'], axis=1)
boxplot(df2)
```

kolom wors area dan mean area di drop dari dataframe df lalu hasilnya disimpan dalam variabel df2. Lalu ditampilkan dalam bentuk boxplot.

```
from sklearn.model_selection import train_test_split
x = df2[df2.columns[:-1]]
y = df2[df2.columns[-1]]
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
stratify=y,random_state=2)
```

baris ini melakukan pemisahan antara variabel independen yang disimpan pada variabel x ditunjukan dengan semua kolom diambil kecuali kolom terakhir. dan variabel dependen yang disimpan pada variabel y yang tunjukan dengan kolom yang diambil hanya kolom terakhir. lalu data di split menjadi data train dan data test dengan rasio data test sebesar 20 persen dengan random_state = 2, nilai ini bebas karena hanya untuk konsistensi data saja ketika digunakan kembali.

```
x_train, x_test = np.asarray(x_train), np.asarray(x_test)
y_train, t_test = np.asarray(y_train), np.asarray(y_test)
```

kesemua x_train, x_test, y_train dany_test dirubah kedalam bentuk numpy array.

```
pad_train = np.zeros(y_train.shape)
```

membuat matrix nol dengan size sesuai dengan y_train

```
y_train = np.stack((y_train, pad_train), axis=-1)
```

menggabungkan dua rray numpy yitu y_train dan pad_train.

```
for i in range(len(y_train)):
if y_train[i][0]==0:
y_train[i][1]=1
else:
y_train[i][1]=0
```

perulangan untuk mengisi jika pada kolom pertama pada baris i sama dengan nol maka kolom kedua akan disi satu jika tidak akan disi nol.

```
network = [
```

```
FlattenLayer(input_shape=(28,)),
FCLayer(28, 32),
ActivationLayer(relu, relu_prime),
FCLayer(32, 64),
ActivationLayer(tanh, tanh_prime),
FCLayer(64, 32),
ActivationLayer(sigmoid, sigmoid_prime),
FCLayer(32, 2),
SoftmaxLayer(2)
]

epochs = 40
learning_rate = 0.1
```

```
# training
for epoch in range(epochs):
error = 0
for x, y_true in zip(x_train, y_train):
# forward
output = x
# print(output.shape, y_true.shape)
for layer in network:
output = layer.forward(output)
# print(output.shape)
```

```
# error (display purpose only)
error += mse(y_true, output)
```

```
# backward
output_error = mse_prime(y_true, output)
for layer in reversed(network):
output_error = layer.backward(output_error, learning_rate)
```

```
error /= len(x_train)
print('%d/%d, error=%f' % (epoch + 1, epochs, error))
```

dilakukan perulangan untuk training data sebanyak epochs = 40 dan learning_rate = 0.1, lalu untuk setiap record pada data x_train dan y_train yang bersesuain akan di train satu persatu dengan looping dengan x dari x_train akan disimpan kepada variabel output, untuk kemudian di proses untuk setiap outputnya ke dalam layer-layer di dalam list network. Lalu hasil output terakhir dari layer akan dihitung errornya terhadap y_true , perhitungan error menggunakan mse (mean square error) . Lalu dilakukan backpropagation mse_prime disimpan kedalam variabel output_error, untuk kemudia setiap ouput error akan diproses secara urutan terbalik dari list network. Lalu error pada setiap epochs akan di cetak.

```
def predict(network, input):
output = input
for layer in network:
output = layer.forward(output)
return output

# [np.argmax(y) == np.argmax(predict(network, x)) for x, y in zip(x_test, y_test)]
for x, y in zip(x_test, y_test):
print(np.argmax(predict(network, x)))
```

dibuat suatu fungsi prediksi dengan inputan berupa network atau layer dan input x. Fungsi ini akan mengembalikan nilai hasil prediksi. Lalu looping dilakukan untuk memprediksi setiap record pada x_test.

```
embddin_dim = len(x_train)
model = tf.keras.Sequential([
tf.keras.layers.Dense(units=28, input_shape=(28,)),
tf.keras.layers.Dense(32, activation='leaky_relu'),
tf.keras.layers.Dense(16, activation='leaky_relu'),
tf.keras.layers.Dense(1, activation='sigmoid')
])
model.summary()
```

pada cell ini code menyimpan banyaknya data x_train kedalam variabel embddin_dim lalu create model dengan library tensor flow dimana layer pertama mempunyain size 28 x 28 , lalu layer kedua dengan activation fungsi leaky_relu dengan banyaknya simpul sebanyak 32, lalu layer 3 dengan fungsi activasi leaku_relu mempunyai simpul 16 dan layer terakhir layer output dengan satu simpul dengan fungsi activasi sigmoid. Lalu model.summary() berfungsi menampilkan informasi tetnang jumlah layer model, output shape dan parameter.

```
print(x_train.shape)
print(y_train.shape)
```

baris code diatas berfungsi untuk mencetak size dari x_train dan y_train

```
x = tf.convert_to_tensor(x, dtype=tf.float32)
y = tf.convert_to_tensor(y, dtype=tf.int64)
```

code diatas berfungsi untuk mengkonversi tipe data variabel x dirubah tipe datanya menjadi float32 dan y menjadi integer64.

```
model.compile(loss='binary_crossentropy', optimizer=RMSprop(lr=0.001), metrics=['accuracy'])
num_epochs = 100
filepath="weights-improvement-{epoch:02d}-{val_accuracy:.2f}.hdf5"
checkpoint = tf.keras.callbacks.ModelCheckpoint(
filepath,
monitor='val_accuracy',
mode='max',
save_best_only=True,
verbose=1)
callbacks_list = [checkpoint]
history = model.fit(x_train, y_train, validation_split=0.33, epochs=num_epochs,
callbacks=callbacks_list, verbose=2)
```

pada baris pertama code diatas model di kompilasi dengan fungsi loss nya binary crossentropy dan mengoptimalkan modelnya degan RMSprop dengan learning rate 0.001 dan ukuran performa model diukur dari akurasinya. lalu num_epochs variabel yang menyimpan nilai 100 untuk banyaknya model nanti di fiting .filepath adalah variabel yang menyimpat path dimana model akan disimpan. Checkpoint adalah callback yang digunakan untuk menyimpan weight selama model dalam pelatihan , weight akan disimpan jika akurasi model lebih baik dari sebelumnya dan hanya menyimpan weight yang terbaik. yang terakhir model.fit adalah tahapan training model dimana bagian data sebanyak 0.33 akan dipakai untuk validasi dan looping sebanyak 100 dan menggunakan callback dari variabel checkpoint.

```
model.load_weights("/content/weights-improvement-88-0.95.hdf5")
y_predict = model.predict(x_test)
```

model di load pada path “/content/weight-improvement-88-0.95.hdf5” . lalu dibuat prediksi dari data x_test.

```
# Evaluate the model on the test data using `evaluate`
print("Evaluate on test data")
results = model.evaluate(x_test, y_test, batch_size=128)
print("test loss, test acc:", results)

# Generate predictions (probabilities -- the output of the last layer)
# on new data using `predict`
print("Generate predictions for 3 samples")
predictions = model.predict(x_test[:3])
print("predictions shape:", predictions)
```

pada cell diatas code mengevalueasi model dengan output berupa loss dan akurasi. lalu pada baigian akhir akan diprediksi 3 data teratas dari x_test lalu hasil prediksi ditampilkan.

```
from keras.datasets import mnist
from keras.utils import np_utils

(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

```
x_train = x_train.astype('float32')
x_train /= 255
y_train = np_utils.to_categorical(y_train)
x_train = x_train[0:1000]
y_train = y_train[0:1000]
```

```
x_test = x_test.astype('float32')
x_test /= 255
y_test = np_utils.to_categorical(y_test)
```

dataset mnist di download disimpan pada variabel x_train, y_train , x_test dan y_test lalu data pada x_train di konversi kedalam bentuk float32. kemudian karena ini adalah gambar hitam-putih maka setiap pixel warna di normalisasi dengan cara dibagi dengan 255, sehingga range nya hanya akan 0 -1 . dibagi 255 karena maksimal warna hitam yaitu 255. lalu data y_train yang merupakan data categorical di encoding kedalam bentuk integer. lalu data x_train dan y_train di slicing sebanyak 1000 data. Yang kemudian data x_test dan y_test di treatment dengan cara yang sama.

```
import matplotlib.pyplot as plt

samples = 10
for test, true in zip(x_test[:samples], y_test[:samples]):
image = np.reshape(test, (28, 28))
plt.imshow(image, cmap='binary')
plt.show()
pred = predict(network, test)[0]
idx = np.argmax(pred)
idx_true = np.argmax(true)
print('pred: %s, prob: %.2f, true: %d' % (idx, pred[idx], idx_true))
```

Pada cell diatas code dimaksudkan untuk menampilkan image dan memprediksi termasuk kategori manakah image tersebut.

4. Pada hasil dataset kanker payudara berdasarkan akurasi dan error selama pelatihan secara garis besar tidak terdapat anomali ataupun keanehan. Akurasi cenderung meningkat pada setiap epoch walaupun untuk beberapa epoch terlihat akurasi menurun cukup jauh namun ini cukup wajar jika data tidak seimbang artinya satu kategori dibanding dengan kategori lainnya terlampau banyak. Atau bisa juga model overfitting artinya model terlalu pintar untuk memprediksi satu kategori dibandung kategori lainnya. Namun selama epoch berjalan sampai akhir kinerja dari model bisa dimprovment dan dipatkan model terbaik hal-hal yang disampaikan sebelumnya tidak terlalu menjadi persoalan.

Sedangkan pada hasil prediksi dataset MNIST berdasarkan error yang didapat selama pelatihan tidak terdapat anomali apapun untuk setiap epoch error semakin kecil yang artinya akurasi semakin baik.