


<p>Nama: (Isi Nama Anda)</p> <p>NIM: (Isi NIM Anda)</p>	 <p>Praktikum Algoritma & Pemrograman</p>	<p>MODUL 1 (Kasus 6)</p> <p>Nama Dosen: (_____)</p>
<p>Hari/Tanggal: Hari, Tanggal Bulan Tahun</p>		<p>Nama Asisten Labratorium: 1. (_____)</p>

Program Sederhana dalam Python

1. Teori Singkat

Pemrosesan data. Pemrosesan data adalah serangkaian langkah yang kita lakukan untuk mengubah data mentah menjadi informasi yang berguna. Data mentah yang kita miliki sering kali tidak rapi dan memiliki banyak masalah, seperti data yang hilang, data duplikat, atau data yang salah label. Proses pemrosesan data mencakup pembersihan, transformasi, dan pengaturan data agar siap untuk analisis lebih lanjut.

• Mislabeled Data

• **Penjelasan:** Data yang diberi label atau kategori yang salah. Ini sering terjadi dalam dataset yang membutuhkan pelabelan manual, seperti klasifikasi gambar atau teks.

• **Dampak:** Mislabeled data dapat mengganggu model pembelajaran mesin karena model dilatih berdasarkan informasi yang salah.

• **Contoh:** Dalam klasifikasi gambar, gambar kucing yang salah dilabeli sebagai anjing.

• Missing Data



•**Penjelasan:** Data yang hilang atau tidak tersedia dalam dataset. Ini dapat terjadi karena kesalahan pengumpulan data, keterbatasan alat pengukuran, atau kesalahan dalam proses entri data.

•**Dampak:** Missing data dapat menyebabkan hasil analisis yang bias atau menyesatkan.

•**Contoh:** Kolom nilai dalam dataset survei yang tidak terisi untuk beberapa responden.

•Corrupted Data

•**Penjelasan:** Data yang rusak dan tidak dapat dibaca atau digunakan dengan benar. Ini bisa disebabkan oleh kesalahan teknis, seperti masalah perangkat keras, perangkat lunak, atau kesalahan manusia selama proses pengolahan data.

•**Dampak:** Corrupted data dapat membuat analisis tidak akurat atau bahkan mustahil untuk dilakukan.

•**Contoh:** File data yang tidak lengkap atau format data yang rusak.

•Outlier Data

•**Penjelasan:** Data yang secara signifikan berbeda dari data lainnya dalam dataset. Outliers bisa terjadi karena kesalahan pengukuran, anomali, atau variasi alami dalam data.

•**Dampak:** Outliers dapat mempengaruhi rata-rata, standar deviasi, dan hasil analisis secara keseluruhan.

•**Contoh:** Dalam analisis pendapatan, satu individu dengan pendapatan jauh lebih tinggi dari yang lain bisa dianggap sebagai outlier.

•Duplicate Data

•**Penjelasan:** Data yang berulang atau tercatat lebih dari sekali. Ini sering terjadi karena kesalahan dalam pengumpulan atau penggabungan data dari berbagai sumber.

•**Dampak:** Duplicate data dapat mengganggu analisis statistik dan mempengaruhi kualitas model.

•**Contoh:** Catatan pelanggan yang sama muncul beberapa kali dalam dataset karena kesalahan penggabungan data.



2. Latihan Pertama

1. Install dan Panggil Packagae yang diperlukan

```
!pip install protobuf==3.20.*
!pip install numpy pandas tensorflow scikit-learn matplotlib seaborn
```

```
import numpy as np
import pandas as pd
import tensorflow as tf

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.utils import plot_model
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler, StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns
from math import sqrt

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

2. Load Data

```
: df = pd.read_csv("kaggle/input/modul-1/Real estate.csv")
df.head()
```

3. Pre-processing data dengan cara menghapus nilai null



Pre-processing Data ¶

Melakukan pra-pemrosesan terhadap data sebelum digunakan. Pada bagian ini akan melakukan "pembersihan" dan "perapihan" data. Tahapan

Cleaning Null Data

Melakukan pengecekan apakah terdapat data nilai kosong. Pada kasus ini data kosong akan dibuang karena data dianggap tidak lengkap.

```
### Sintaks ini berguna untuk mengecek jumlah nilai null dari setiap atribut data.
### Jika terdapat nilai null, maka akan ditampilkan jumlahnya.
### Jika tidak terdapat nilai null, maka akan ditampilkan nilai 0.

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

4. Membuat Heatmap

```
### Jika terdapat nilai null maka gunakan sintaks dibawah berikut.

df = df.dropna()
```

Drop Columns

Pada bagian ini, data yang tidak terpakai akan dilakukan pembuangan.

```
df = df.drop(columns = ['No', 'X1 transaction date'])
```

Create HeatMap

Pada bagian ini akan melakukan analisa mengenai hubungan antara fitur dengan target. Hal ini berguna untuk membuang fitur yang tidak berhubungan dengan target dan yang berpotensi untuk memngganggu pembelajaran model.

```
dataplot = sns.heatmap(df.corr(), cmap="YlGnBu", annot=True)
plt.show()
```

5. Membagi data menjadi

Split Feature and Target

Pada bagian ini akan memisahkan data yang menjadi ciri rumah dan nilai yang akan digunakan sebagai target prediksi. Pada kasus ini, harga rumah menjadi target prediksi, maka akan dilakukan pemisahan. Pada kasus ini, berdasarkan heatmap, fitur yang akan diambil adalah **X3 distance to the nearest MRT station** dan **X4 number of convenience stores**

```
X = df[[
    # "X1 transaction date",
    # "X2 house age",
    "X3 distance to the nearest MRT station",
    "X4 number of convenience stores",
    # "X5 latitude",
    # "X6 longitude"
]].copy()
Y = df["Y house price of unit area"].copy()

print(X.columns.values)
print(Y)
```



6. Min-Max Scaller

```
X['X3 distance to the nearest MRT station'] = MinMaxScaler().fit_transform(np.sqrt(X['X3 distance to the nearest MRT station'].values).reshape(-1, 1))
X['X4 number of convenience stores'] = MinMaxScaler().fit_transform(np.sqrt(X['X4 number of convenience stores'].values).reshape(-1, 1))
Y = MinMaxScaler().fit_transform(np.sqrt(Y.values).reshape(-1, 1))
```

7. Membuat fungsi untuk model scaler

```
### Terdapat 3 layers.
### Layer pertama adalah Input Layer yang didefinisikan pada input_shape
### Layer kedua adalah Hidden Layer yang didefinisikan pada fungsi Dense pertama dengan jumlah node 8
### Layer ketiga adalah Output Layer yang didefinisikan pada fungsi Dense kedua dengan jumlah node 1

def create_model():
    model = Sequential([
        Dense(1, input_shape=(X.columns.shape[0],)),
        ### input shape berfungsi untuk menentukan jumlah masukkan fitur
    ])

    ### Compile model berguna untuk mengisi optimizer, loss, dan metrik evaluasi.

    model.compile(optimizer='adam', loss='mse')

    return model
```

8. Kasus Pertama (Melatih model dengan ratio 70:30)



Splitting Data Between Train Data and Test Data (70:30)

Pada bagian ini akan melakukan pembagian data latih dan data uji. Selayaknya mahasiswa yang memerlukan bahan belajar dan ujian. Analisa data latih terletak sebagai data belajar dan data uji sebagai soal ujian.

Data latih berguna untuk memberikan pembelajaran pada model AI yang akan dilatih dan data uji berguna untuk menguji kemampuan model AI yang telah melalui proses pembelajaran.

Ratio pembagian akan lebih besar pada data latih dibandingkan data data uji. Pada kasus ini, ratio akan berupa 70:30

```
1: ### Pemisahan data Latih dan data uji dengan fungsi train_test_split().
### Parameter :
### frac      = menentukan ratio jumlah data Latih yang akan diambil.
### random_state = berfungsi untuk menghasilkan hasil acak yang sama jika
###            diinput dengan angka yang sama.
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, train_size=0.7)
```

Build Prediction Model

```
1: model_kasus_1 = create_model()

1: ### Untuk melihat hasil dari model yang telah dibangun dapat dilakukan dengan model summary
### Untuk hasil berupa gambar dapat menggunakan fungsi plot_model

model_kasus_1.summary()
plot_model(model_kasus_1, show_dtype=True, show_layer_names=True, show_shapes=True)
```

9. Train and Evaluate Model

Pada bagian akan menjelaskan bagaimana cara model dilatih dan dievaluasi kinerjanya.

Train Model

Pada bagian ini akan melakukan pelatihan terhadap model AI. Fungsi Fit() berguna untuk melatih model dengan data latih.

Parameter yang wajib diisi:

Fitur Data Latih = X_Train

Target Data Latih = Y_Train

Epochs = Berapa kali perputaran melatih model

Verbose = Keluaran output proses pelatihan

```
hist = model_kasus_1.fit(X_train, Y_train, epochs=500, verbose=1)
```

10. Membuat plot untuk history epoch



```
### Melihat perjalanan dari loss pada setiap epoch

plt.plot(hist.history['loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.show()
```

11. Evaluasi Model

```
pred = model_kasus_1.predict(X_test)
result_70_30 = mean_squared_error(Y_test, pred)
print(result_70_30)
```

12. Kasus Kedua (Melatih model dengan ratio 80:20)

Kasus Kedua (Melatih model dengan ratio 80:20)

Pada kasus kedua ini akan mengulang langkah pada kasus pertama, tetapi ratio pembagian data latih dan data uji kita ubah menjadi 80:20

Splitting Data Between Train Data and Test Data (80:20)

Pada bagian ini akan melakukan pembagian data latih dan data uji dengan ratio 80:20

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, train_size=0.8)
```

Build Prediction Model

```
model_kasus_2 = create_model()
```

```
model_kasus_2.summary()
plot_model(model_kasus_2, show_dtype=True, show_layer_names=True, show_shapes=True)
```

```
hist = model_kasus_2.fit(X_train, Y_train, epochs=500, verbose=1)
```



13. Membuat plot untuk history epoch

```
### Melihat perjalanan dari loss pada setiap epoch

plt.plot(hist.history['loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.show()
```

14. Evaluasi Model

```
pred = model_kasus_2.predict(X_test)
result_80_20 = mean_squared_error(Y_test, pred)
print(result_80_20)
```

15. Model Performance Analysis

Lakukan analisa berdasarkan dari hasil kedua kasus

```
print("Kasus 1 (70:30):",result_70_30)
print("Kasus 2 (80:20):",result_80_20)
```

3. Kesimpulan

1. Kita dapat mengetahui... (Tolong Isi lebih dari dua baris!)



4. Tugas Kelas

Tugas

Jika terdapat data mengenai penerbangan Jakarta-Tokyo berikut ini :

Nama Pesawat	Harga	Lama Penerbangan	Transit (Stop)	Voucher Hotel
NAM	3.200.000	41 jam 30 menit	2	Ya
Air Asia	3.300.000	35 jam	2	Tidak
EVA Air	3.900.000	20 jam 5 menit	1	Tidak
Cathay Pasific	5.300.000	14 jam 25 menit	1	Tidak
Cathay Pasific	6.500.000	15 jam 15 menit	1	Tidak
Malaysia Airlines	6.600.000	21 jam 40 menit	1	Ya
Garuda	6.300.000	7 jam 25 menit	0	Ya
Garuda	9.500.000	7 jam 55 menit	0	Ya
ANA	26.000.000	7 jam 55 menit	0	Ya
ANA	27.000.000	7 jam	0	Ya

lakukan normalisasi dengan Min-Max, Decimal Scaling, dan Z-Score, lalu berikan analisa anda terhadap hasil yang didapat!

5. Cek List (✓)

No	Elemen Kompetensi	Penyelesaian	
		Selesai	Tidak Selesai
1.	Latihan Pertama		
2.	Tugas Kelas		

6. Formulir Umpan Balik

No	Elemen Kompetensi	Waktu Pengerjaan	Kriteria
1.	Latihan Pertama	... Menit	...
2.	Tugas Kelas		



Keterangan:

1. Menarik
2. Baik
3. Cukup
4. Kurang

Penanggung Jawab Praktikum

(_____)

Kepala Lab. Praktikum

(_____)

