

# Read Dataset

Pustaka `pandas` digunakan untuk membaca dan menganalisa data dalam berbagai format. Secara umum fungsi-fungsi pada `pandas` mirip dengan `Microsoft Excel`. Pada `Microsoft Excel` proses analisa menggunakan GUI (*Graphical User Interface*), pada `pandas` proses analisa menggunakan pendekatan *programming*.

Informasi lengkap mengenai `pandas` dapat dilihat di <https://pandas.pydata.org/>.

```
In [1]: import pandas as pd
```

```
df = pd.read_csv(filepath_or_buffer="../datasets/titanic_numerical_features.csv", index_
df.head(n=5)
```

```
Out[1]:
```

	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId						
1	0	3	22.0	1	0	7.2500
2	1	1	38.0	1	0	71.2833
3	1	3	26.0	0	0	7.9250
4	1	1	35.0	1	0	53.1000
5	0	3	35.0	0	0	8.0500

```
In [2]: df.tail(n=5)
```

```
Out[2]:
```

	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId						
887	0	2	27.0	0	0	13.00
888	1	1	19.0	0	0	30.00
889	0	3	NaN	1	2	23.45
890	1	1	26.0	0	0	30.00
891	0	3	32.0	0	0	7.75

## Dataset Information

Untuk mengakses informasi pada `DataFrame` seperti jumlah baris, jumlah kolom, tipe data kolom, ukuran memori, dan lain-lain dapat menggunakan perintah `DataFrame.info()`.

```
In [3]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 891 entries, 1 to 891
Data columns (total 6 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Survived    891 non-null    int64
```

```
1  Pclass      891 non-null    int64
2  Age         714 non-null    float64
3  SibSp       891 non-null    int64
4  Parch       891 non-null    int64
5  Fare        891 non-null    float64
dtypes: float64(2), int64(4)
memory usage: 48.7 KB
```

## Quick EDA (Exploratory Data and Analysis)

Salah satu produk turunan dari `pandas` adalah `pandas-profiling`. Pustaka `pandas-profiling` menyediakan fungsi untuk melakukan EDA (Exploratory Data and Analysis) dengan cepat. Menampilkan *overview, variables, interactions, correlations, missing value*, dan *samples* dalam bentuk laporan berformat *.html*.

Informasi mengenai `pandas-profiling` dapat dilihat di <https://pandas-profiling.ydata.ai/docs/master/index.html>.

```
In [4]: from pandas_profiling import ProfileReport

profile = ProfileReport(df=df, title="Pandas Profiling Report")
profile

Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]
Generate report structure:  0%|          | 0/1 [00:00<?, ?it/s]
Render HTML:      0%|          | 0/1 [00:00<?, ?it/s]
```

# Overview

[Overview](#)[Alerts](#) **16**[Reproduction](#)

## Dataset statistics

Number of variables	7
Number of observations	891
Missing cells	177
Missing cells (%)	2.8%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	48.9 KiB
Average record size in memory	56.1 B

## Variable types

Numeric	5
Categorical	2

# Variables

Out[4]:

## Drop Missing Value

Tahapan *drop missing value* sangat penting dilakukan agar fitur yang digunakan pada saat *training* tidak ada yang kosong. Tahapan ini perlu pertimbangan yang memadai, karena bisa saja menghilangkan suatu fitur dapat mengurangi informasi dari *dataset* yang dapat mempengaruhi performa dari model.

In [5]: `df.isna().sum()``Survived` 0

```
Out[5]: Pclass      0
Age      177
SibSp     0
Parch     0
Fare      0
dtype: int64
```

```
In [6]: df = df.dropna()
df.isna().sum()
```

```
Out[6]: Survived      0
Pclass      0
Age          0
SibSp        0
Parch        0
Fare         0
dtype: int64
```

## Dataset Splitting

Pemisahan *dataset* menjadi *train* dan *test set* dilakukan untuk mengurangi *data leakage* atau kebocoran informasi. Model tidak boleh mengetahui pola dari *test set*. Untuk melakukan pemisahan *dataset* dapat menggunakan pustaka `scikit-learn`, yaitu fungsi `train_test_split`.

```
In [7]: from sklearn.model_selection import train_test_split

X = df.drop(columns="Survived")
y = df.Survived

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y)

print(f"X_train shape, rows: {X_train.shape[0]}, columns: {X_train.shape[1]}")
print(f"X_train shape, rows: {y_train.shape[0]}")
print(f"X_test shape, rows: {X_test.shape[0]}, columns: {X_test.shape[1]}")
print(f"y_test shape, rows: {y_test.shape[0]}")

X_train shape, rows: 571, columns: 5
X_train shape, rows: 571
X_test shape, rows: 143, columns: 5
y_test shape, rows: 143
```

## Train Model

Untuk membuat model *machine learning* dapat menggunakan pustaka `scikit-learn` dan mengimpor algoritma yang diinginkan. `scikit-learn` menyediakan berbagai fungsionalitas dan algoritma terkait *machine learning*.

Untuk lebih lengkapnya terkait fungsi dan algoritma yang disediakan `scikit-learn` dapat mengunjungi <https://scikit-learn.org/stable/modules/classes.html#>.

```
In [8]: from sklearn.neighbors import KNeighborsClassifier

model = KNeighborsClassifier(n_neighbors=3)
model.fit(X=X_train, y=y_train)
```

```
Out[8]: KNeighborsClassifier(n_neighbors=3)
```

# Evaluate Model

Evaluasi dilakukan untuk meninjau performa dari model. Pada kasus klasifikasi metrik evaluasi yang digunakan, antara lain akurasi, *precision*, *recall*, *f1-score*, dan lain-lain. Pengujian dapat dilakukan dengan menggunakan *train* dan *test dataset*. Tetapi, pengujian dengan *test dataset* menjadi acuan utama dalam meninjau performa model.

Untuk lebih lengkapnya dapat mengunjungi halaman berikut <https://scikit-learn.org/stable/modules/classes.html#sklearn-metrics-metrics>.

```
In [9]: from sklearn.metrics import classification_report

y_train_preds = model.predict(X=X_train)
print(classification_report(y_true=y_train, y_pred=y_train_preds))
```

	precision	recall	f1-score	support
0	0.82	0.86	0.84	339
1	0.79	0.72	0.75	232
accuracy			0.81	571
macro avg	0.80	0.79	0.80	571
weighted avg	0.81	0.81	0.81	571

```
In [10]: y_test_preds = model.predict(X=X_test)
print(classification_report(y_true=y_test, y_pred=y_test_preds))
```

	precision	recall	f1-score	support
0	0.71	0.72	0.71	85
1	0.58	0.57	0.57	58
accuracy			0.66	143
macro avg	0.64	0.64	0.64	143
weighted avg	0.66	0.66	0.66	143

## Predict use New Dataset

Setelah melakukan *training* dan evaluasi terhadap model, model dapat digunakan untuk melakukan prediksi menggunakan data baru.

```
In [11]: df_new = pd.DataFrame(
    data=[[1, 34., 2, 7, 300.],
          [3, 50., 0, 0, 7.34]],
    columns=X_train.columns,
    index=[892, 893]
)

new_preds = model.predict(X=df_new)

df_new["Survived Prediction"] = new_preds
df_new
```

```
Out[11]:
```

	Pclass	Age	SibSp	Parch	Fare	Survived Prediction
892	1	34.0	2	7	300.00	1

# Save Model

Setelah *training* dan evaluasi model, model dapat disimpan dengan menggunakan pustaka `joblib`. Fungsi `dump` untuk menyimpan objek model dan `load` untuk menggunakan objek yang telah disimpan sebelumnya.

```
In [12]: from joblib import dump, load

dump(value=model, filename="../pretrained_models/model_simple_workflow.joblib")
model = load(filename="../pretrained_models/model_simple_workflow.joblib")
model.predict(df_new.iloc[:, 0:5])
```

```
Out[12]: array([1, 0], dtype=int64)
```

---

Semoga bermanfaat yah 😊

Dibuat dengan penuh ❤ oleh [haloapping](#)

---