Read Dataset

Untuk membaca dan menganalisa dataset digunakan `pandas`. Pada `pandas` terdapat fungsi `read_csv()` untuk membaca keseluruhan isi dari berkas berformat .csv, `head()` untuk menampilkan *n* baris pertama data, dan `tail()` untuk menampilkan *n* baris terakhir data.

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

df = pd.read_csv(filepath_or_buffer="../datasets/titanic_numeric_and_categoric_features.
```

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

Out[2]:

PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
1	0	3	male	22.0	1	0	7.2500	S
2	1	1	female	38.0	1	0	71.2833	C
3	1	3	female	26.0	0	0	7.9250	S
4	1	1	female	35.0	1	0	53.1000	S
5	0	3	male	35.0	0	0	8.0500	S

In [3]:
```
df.tail(n=5)
```

Out[3]:

PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
887	0	2	male	27.0	0	0	13.00	S
888	1	1	female	19.0	0	0	30.00	S
889	0	3	female	NaN	1	2	23.45	S
890	1	1	male	26.0	0	0	30.00	C
891	0	3	male	32.0	0	0	7.75	Q

Dataset Information

Informasi dari dataset, seperti jumlah baris (sampel), jumlah kolom, nama kolom, tipe data setiap kolom, ruang penyimpanan, dan lain-lain dapat diakses dengan menggunakan fungsi `info()`.

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 891 entries, 1 to 891
Data columns (total 8 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  891 non-null    int64
 1   Pclass    891 non-null    int64
 2   Sex       891 non-null    object
```

```
 3   Age         714 non-null    float64
 4   SibSp       891 non-null    int64
 5   Parch       891 non-null    int64
 6   Fare        891 non-null    float64
 7   Embarked    889 non-null    object
dtypes: float64(2), int64(4), object(2)
memory usage: 62.6+ KB
```

Quick EDA (Exploratory Data and Analysis)

`pandas profiling` menyediakan fungsionalitas untuk melakukan EDA (*Exploratory Data and Analysis*) dengan cepat. Laporan akan dibuat dalam format .html.

In [5]:
```
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 21 | Reproduction

Dataset statistics

Number of variables	9
Number of observations	891
Missing cells	179
Missing cells (%)	2.2%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	62.8 KiB
Average record size in memory	72.1 B

Variable types

Numeric	5
Categorical	4

Variables

Out[5]:

Dataset Splitting

Proses pemisahan dataset dapat dilakukan dengan menggunakan fungsi `train_test_split` yang ada pada `scikit-learn`.

In [6]:
```
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 : {X_train.shape}")
print(f"X_train shape : {y_train.shape}")
print(f"X_test shape  : {X_test.shape}")
print(f"y_test shape  : {y_test.shape}")
```

```
X_train shape : (712, 7)
X_train shape : (712,)
X_test shape  : (179, 7)
y_test shape  : (179,)
```

Build Model

Pada *notebook* ini model dibuat dengan menggunakan mekanisme *pipeline*. *Pipeline* adalah salah satu modul pada `scikit-learn` yang berfungsi untuk membungkus setiap tahapan yang biasanya dilakukan secara terpisah pada saat membuat model *machine learning*. Dengan menggunakan modul *pipeline* kode akan menjadi lebih rapi dan ringkas, sehingga mudah untuk dikelola dan dikembangkan bila ada kasus tertentu yang mengharuskan mengubah alur dari kode. Tujuan lain dari penggunaan *pipeline* adalah mengurangi *data leakage* (kebocoran informasi). Seperti yang diketahui pada saat proses *training model*, model harus dipastikan tidak mengetahui pola atau informasi dari dataset pengujian. Dengan penggunaan *pipeline* skenario tersebut dapat dilakukan dengan mudah.

Numerical Preprocessor Pipeline

Pipeline ini berfungsi untuk menangani kolom yang bertipe numerikal (continous value).

In [7]:
```
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import MinMaxScaler

numerical_prep_pipeline = Pipeline(
    steps=[
        ("step1_imputer", SimpleImputer(strategy="mean")),
        ("step2_scaler", MinMaxScaler())
    ]
)
```

Categorical Preprocessor Pipeline

Pipeline ini berfungsi untuk menangani kolom yang bertipe kategorikal (discrete value).

In [8]:
```
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder

categorical_prep_pipeline = Pipeline(
    steps=[
        ("step1_imputer", SimpleImputer(strategy="most_frequent")),
        ("step2_encoder", OneHotEncoder())
    ]
)
```

Data Preprocessor Pipeline

Pipeline ini merupakan pembungkus untuk dua *pipeline* sebelumnya, yaitu *numerical pipeline* dan *categorical pipeline*. Pipeline ini akan memilah kolom berdasarkan tipe datanya dan diteruskan ke *pipeline* yang sesuai dengan kategorinya.

In [9]:
```
from sklearn.compose import ColumnTransformer, make_column_selector

preprocess_pipeline = ColumnTransformer(
    transformers=[
        ("step1_numerical_pipeline", numerical_prep_pipeline, make_column_selector(dtype
        ("step2_categorical_pipeline", categorical_prep_pipeline, make_column_selector(d
    ]
)
```

Model Pipeline

Pipeline ini merupakan pembungkus paling akhir dari semua *pipeline* yang telah dibuat sebelumnya. Pada *pipeline* ini terdiri dari 2 langkah, yaitu yang pertama transformasi fitur dan langkah kedua adalah melakukan *training model*

In [10]:
```
from sklearn.neighbors import KNeighborsClassifier

model_pipeline = Pipeline([
    ("step1_preprocess_pipeline", preprocess_pipeline),
    ("step2_algo", KNeighborsClassifier())
])
```

Feature Transform

Karena pada pipeline sebelumnya sudah termasuk pipeline transformasi fitur, maka transformasi fitur dapat diakses melalui pipeline transformasi fitur dengan mengakses fungsi `fit_transform()`.

In [11]:
```
df_transform = pd.DataFrame(preprocess_pipeline.fit_transform(X=X_train))
df_transform[:5]
```

Out[11]:

	0	1	2	3	4	5	6	7	8	9
0	0.5	0.220910	0.000	0.000000	0.022447	0.0	1.0	0.0	0.0	1.0
1	0.0	0.373592	0.000	0.000000	0.444099	0.0	1.0	1.0	0.0	0.0
2	1.0	0.409399	0.000	0.000000	0.015176	0.0	1.0	0.0	0.0	1.0
3	0.0	0.447097	0.125	0.333333	0.234224	0.0	1.0	0.0	0.0	1.0
4	1.0	0.346569	0.000	0.000000	0.018543	0.0	1.0	0.0	0.0	1.0

Train Model with Grid Search CV Scenario

In [12]:
```
from sklearn.model_selection import GridSearchCV

parameters = {
    "step2_algo__n_neighbors": range(1, 51, 2),
```

```
    "step2_algo__weights": ["uniform", "distance"],
    "step2_algo__p": [1, 2]
}

model = GridSearchCV(estimator=model_pipeline, param_grid=parameters, cv=5, scoring="acc
model.fit(X_train, y_train)
pd.DataFrame(model.cv_results_).sort_values(by="rank_test_score").iloc[:5, :]
```

Out[12]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_step2_algo__n_neighbors	param_step2_alg
48	0.043257	0.002120	0.042342	0.003255	25	
44	0.035227	0.007730	0.038746	0.008152	23	
52	0.030016	0.002643	0.030387	0.002324	27	
50	0.037822	0.003457	0.038241	0.005676	25	
38	0.031476	0.002641	0.030833	0.003399	19	

Evaluate Model

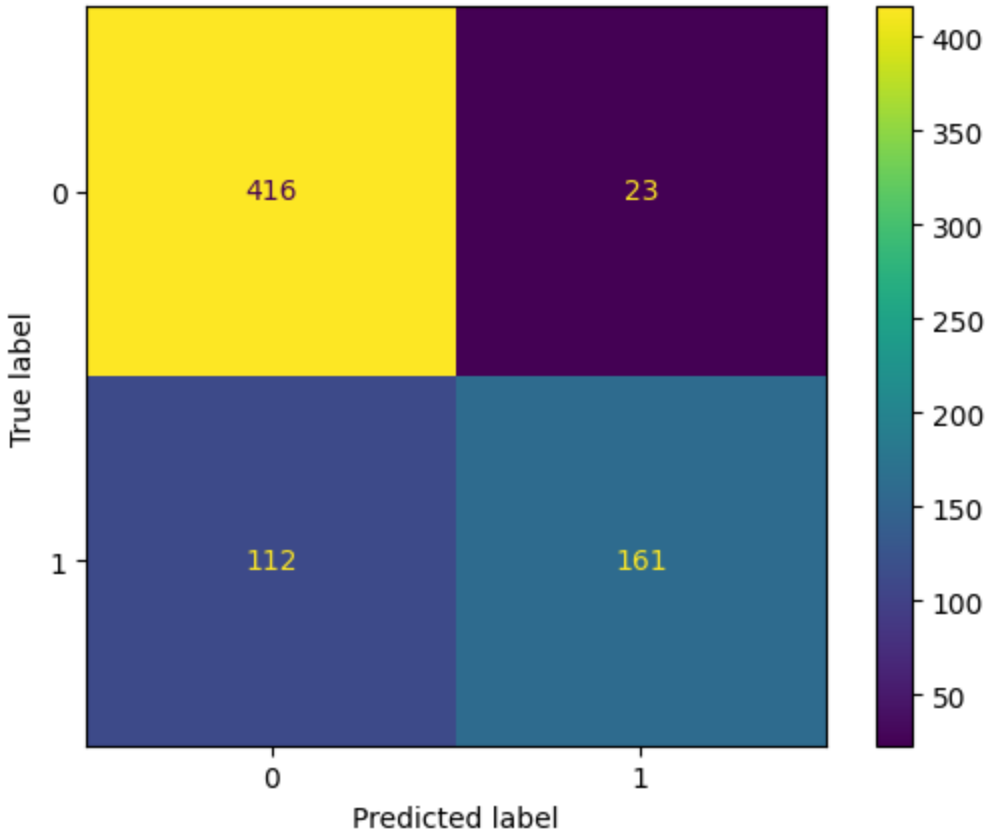
Pada `scikit-learn` telah tersedia rangkuman metrik dari klasifikasi yang dapat diakses menggunakan fungsi `classification_report`. Pada fungsi tersebut berisi rangkuman akurasi, *precision*, *recall*, *f1-score*, dan lain-lain untuk setiap kelas.

Train Dataset

In [13]:
```
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       0.79      0.95      0.86       439
           1       0.88      0.59      0.70       273

    accuracy                           0.81       712
   macro avg       0.83      0.77      0.78       712
weighted avg       0.82      0.81      0.80       712
```

In [14]:
```
import matplotlib.pyplot as plt

from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

cm = confusion_matrix(y_true=y_train, y_pred=y_train_preds, labels=model.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=model.classes_)
disp.plot();
```

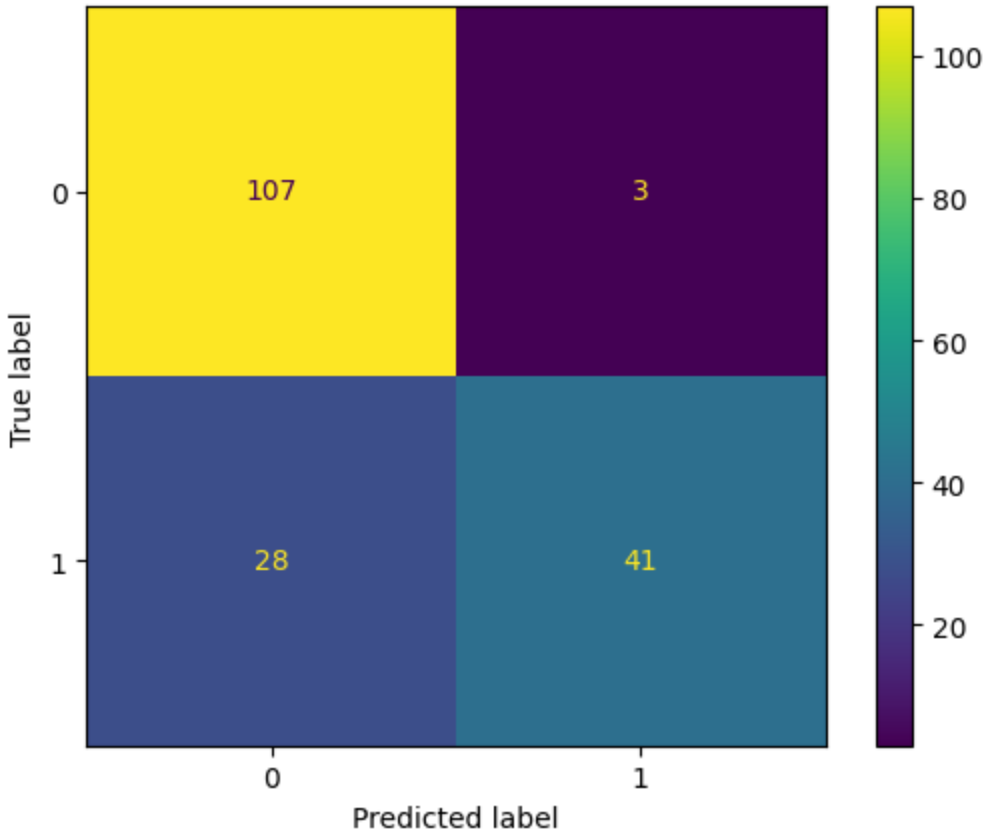


Test Dataset

In [15]:
```
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       0.79      0.97      0.87       110
           1       0.93      0.59      0.73        69

    accuracy                           0.83       179
   macro avg       0.86      0.78      0.80       179
weighted avg       0.85      0.83      0.82       179
```

In [16]:
```
import matplotlib.pyplot as plt

from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

cm = confusion_matrix(y_true=y_test, y_pred=y_test_preds, labels=model.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=model.classes_)
disp.plot();
```



Predict use New Dataset

Prediksi dapat dilakukan dengan mengakses fungsi `predict()` pada objek model yang telah dilatih.

In [17]:
```
df_new = pd.DataFrame(
    data=[[1, "male", 34., 2, 7, 300., "S"],
          [3, "female", 50., 0, 0, 7.34, "Q"]],
    columns=X_train.columns,
    index=[892, 893]
)

new_preds = model.predict(X=df_new)

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

Out[17]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Survived Predicition
892	1	male	34.0	2	7	300.00	S	0
893	3	female	50.0	0	0	7.34	Q	1

Save Prediction

In [18]:
```
from datetime import datetime

now = datetime.now()
now = now.strftime("%d_%m_%Y-%H_%M_%S")

df_new.to_csv("../predictions/" + now + "_complex_workflow.csv", index=False)
```

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 [19]:
```
from datetime import datetime
from joblib import dump, load

now = datetime.now()
now = now.strftime("%d_%m_%Y-%H_%M_%S")
dump(value=model, filename="../pretrained_models/" + now + "_complex_workflow.joblib")
model = load(filename="../pretrained_models/" + now + "_complex_workflow.joblib")
model.predict(df_new.iloc[:, 0:7])
```

Out[19]:
```
array([0, 1], dtype=int64)
```

Semoga bermanfaat yah 😁

Dibuat dengan penuh 🤍 oleh haloapping
