J.62DMI00.010.1

Modul 7: Menentukan Label Data Menggunakan Python

Modul ini memperkenalkan teknik penentuan label data menggunakan Python. Pelabelan data merupakan langkah penting dalam siklus hidup data science yang menentukan kualitas model machine learning yang akan dibangun. Melalui modul ini, Anda akan mempelajari berbagai teknik untuk menentukan dan mengaplikasikan label pada dataset, termasuk pelabelan manual, pelabelan otomatis, dan validasi hasil pelabelan.

Perangkat Pendukung

No.	Sistem	Keterangan
1	Python versi 3.7 atau lebih tinggi	Untuk kebutuhan kompilasi kode
2	Jupyter Notebook atau Google Colab	Sebagai alat kompilasi python berbasis cloud
3	Library Python: pandas, numpy, scikit-learn, matplotlib, seaborn	Untuk analisis, manipulasi, labeling, dan visualisasi data

Dataset yang Digunakan

Dalam modul ini, kita akan menggunakan dataset berikut yang tersedia melalui repository:

1. Data Mahasiswa

   ```
   https://media.githubusercontent.com/media/evanightly/codeasy/refs/heads/main/fastapi/datasource/Data%20Mahasiswa/Data%20mahasiswa.csv
   ```

2. Data Student Depression Dataset

   ```
   https://media.githubusercontent.com/media/evanightly/codeasy/refs/heads/main/fastapi/datasource/Data%20Student%20Depression%20Dataset/student_depression_dataset.csv
   ```

3. Data Student Performance and Behavior Dataset

   ```
   https://media.githubusercontent.com/media/evanightly/codeasy/refs/heads/main/fastapi/datasource/Data%20Student%20Performance%20and%20Behavior%20Dataset/Students_Grading_Dataset.csv
   ```

Pengenalan Library untuk Pelabelan dan Pemrosesan Data

Sebelum kita mulai, mari kenali beberapa library dan fungsi yang akan sering digunakan dalam pelabelan data:

1. Scikit-learn (Preprocessing)

Scikit-learn menyediakan berbagai tools untuk preprocessing data, termasuk pelabelan dan transformasi kategorikal:

a. LabelEncoder

```
from sklearn.preprocessing import LabelEncoder
```

Metode	Deskripsi	Parameter	Return Type
`fit(y)`	Mempelajari mapping label dari dataset	y (array-like): Label target	`self`
`transform(y)`	Mengubah label menjadi nilai numerik normalisasi	y (array-like): Label target	array dengan label yang telah dienkode
`fit_transform(y)`	Menggabungkan fit() dan transform()	y (array-like): Label target	array dengan label yang telah dienkode
`inverse_transform(y)`	Mengembalikan label asli	y (array-like): Nilai numerik yang akan diubah ke label asli	array dengan label asli
`classes_`	Array kelas	-	array dengan kelas unik yang ditemukan

Contoh: `encoder = LabelEncoder(); encoded_labels = encoder.fit_transform(df['Gender'])` akan mengubah label kategorikal dalam kolom 'Gender' menjadi nilai numerik (misalnya 'Laki-laki' → 0, 'Perempuan' → 1).

Dokumentasi: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html

b. OneHotEncoder

```
from sklearn.preprocessing import OneHotEncoder
```

Parameter	Tipe Data	Deskripsi	Wajib/Opsional	Default
`categories`	'auto' atau list	Kategori untuk setiap fitur	Opsional	'auto'
`drop`	'first', 'if_binary' atau None	Kategori yang akan di-drop	Opsional	None
`sparse`	bool	Apakah akan mengembalikan sparse matrix	Opsional	True
`handle_unknown`	'error', 'ignore'	Tindakan jika kategori unseen ditemukan	Opsional	'error'

Contoh: `encoder = OneHotEncoder(sparse=False); encoded = encoder.fit_transform(df[['Program']])` akan mengubah kolom 'Program' menjadi format one-hot encoding, di mana setiap nilai kategori akan menjadi kolom baru dengan nilai 0 atau 1.

Dokumentasi: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html

2. Pandas (Kategorikal)

Pandas juga menyediakan metode untuk mengelola variabel kategorikal:

a. pandas.cut()

```
pd.cut(x, bins, labels=None, right=True, include_lowest=False)
```

Parameter	Tipe Data	Deskripsi	Wajib/Opsional	Default
`x`	array-like	Data 1-dimensi yang akan dibagi	Wajib	-
`bins`	int, sequence	Jumlah bin atau batas bin	Wajib	-
`labels`	array atau bool	Nama untuk bin yang dihasilkan	Opsional	None
`right`	bool	Interval tertutup di kanan	Opsional	True
`include_lowest`	bool	Apakah interval pertama mencakup nilai terendah	Opsional	False

Contoh: `pd.cut(df['Nilai'], bins=[0, 60, 75, 85, 100], labels=['D', 'C', 'B', 'A'])` akan mengkategorikan nilai numerik dalam kolom 'Nilai' menjadi kategori A, B, C, D berdasarkan interval yang ditentukan.

Dokumentasi: https://pandas.pydata.org/docs/reference/api/pandas.cut.html

b. pandas.qcut()

```
pd.qcut(x, q, labels=None, precision=3)
```

Parameter	Tipe Data	Deskripsi	Wajib/Opsional	Default
`x`	array-like	Data 1-dimensi yang akan dibagi	Wajib	-
`q`	int atau array	Jumlah kuantil atau daftar kuantil	Wajib	-
`labels`	array atau None	Label untuk bin yang dihasilkan	Opsional	None
`precision`	int	Presisi untuk batas kuantil	Opsional	3

Contoh: `pd.qcut(df['Nilai'], q=4, labels=['Quartile 1', 'Quartile 2', 'Quartile 3', 'Quartile 4'])` akan membagi nilai dalam kolom 'Nilai' menjadi 4 kuartil dengan label yang ditentukan.

Dokumentasi: https://pandas.pydata.org/docs/reference/api/pandas.qcut.html

c. pandas.get_dummies()

```
pd.get_dummies(data, prefix=None, prefix_sep='_', dummy_na=False, drop_first=False)
```

Parameter	Tipe Data	Deskripsi	Wajib/Opsional	Default
`data`	array-like, Series, DataFrame	Data yang akan di-dummy-kan	Wajib	-
`prefix`	str, list, atau dict	Nama prefiks kolom hasil	Opsional	None
`prefix_sep`	str	Separator antara prefiks dan nilai	Opsional	'_'
`dummy_na`	bool	Tambahkan kolom untuk NaN	Opsional	False
`drop_first`	bool	Drop kolom pertama	Opsional	False

Contoh: `pd.get_dummies(df['Program'], prefix='Program')` akan mengubah kolom 'Program' menjadi beberapa kolom dummy (one-hot encoding) dengan nama depan 'Program_'.

Dokumentasi: https://pandas.pydata.org/docs/reference/api/pandas.get_dummies.html

Materi Pembelajaran

1. Pengenalan Label Data

Label data adalah proses memberikan identitas atau kategori pada data, yang sangat penting dalam machine learning terutama untuk supervised learning. Tahap ini menentukan target yang ingin diprediksi oleh model. Mari kita lihat contoh penggunaan beberapa metode pelabelan:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder

# Memuat data
df_student =
pd.read_csv('https://media.githubusercontent.com/media/evanightly/codeasy/refs/heads/main/fastapi/datasource/Data%20Student%20Depression%20Dataset/student_depression_dataset.csv')
```

```
# Melihat 5 baris pertama
print(df_student.head())

# Informasi struktur dataset
print(df_student.info())

# Melihat kolom kategorikal yang perlu dilabel
categorical_cols =
df_student.select_dtypes(include=['object']).columns.tolist()
print("Kolom kategorikal:", categorical_cols)
```

Output: Kode di atas akan menampilkan informasi tentang dataset depresi mahasiswa, termasuk 5 baris pertama, struktur dataset (seperti tipe data setiap kolom), dan daftar kolom kategorikal yang mungkin memerlukan pelabelan. Output menunjukkan bahwa kolom seperti 'Gender', 'City', 'Profession', dll. adalah kategorikal dan bisa dilabel untuk analisis lebih lanjut.

2. Teknik Pelabelan Kategorikal dengan LabelEncoder

LabelEncoder berguna untuk mengubah label kategorikal menjadi nilai numerik:

```
# Membuat LabelEncoder
le = LabelEncoder()

# Mengaplikasikan LabelEncoder pada kolom Gender
df_student['Gender_Encoded'] = le.fit_transform(df_student['Gender'])

# Melihat hasil encoding
print(df_student[['Gender', 'Gender_Encoded']].head(10))

# Melihat mapping yang dibuat oleh LabelEncoder
print("Mapping kelas:", list(zip(le.classes_, range(len(le.classes_)))))
```

Output: Kode ini mengubah label kategorikal di kolom 'Gender' (seperti 'Male', 'Female') menjadi nilai numerik (0, 1). Output menampilkan tabel perbandingan nilai asli dan hasil encoding, serta mapping yang dibuat (misalnya 'Female' → 0, 'Male' → 1). Teknik ini menyederhanakan data kategorikal untuk pemrosesan algoritma machine learning yang membutuhkan input numerik.

3. One-Hot Encoding untuk Variabel Kategorikal

One-hot encoding lebih sesuai untuk variabel kategorikal nominal (tidak memiliki urutan tertentu):

```
# One-hot encoding dengan pandas
profession_dummies = pd.get_dummies(df_student['Profession'], prefix='Prof')

# Tambahkan hasil encoding ke dataframe
df_with_dummies = pd.concat([df_student[['id', 'Gender', 'Age']],
profession_dummies], axis=1)
```

```
# Lihat hasilnya
print(df_with_dummies.head())
```

Output: Kode ini mengubah kolom 'Profession' menjadi beberapa kolom baru, di mana setiap profesi menjadi kolom terpisah dengan nilai 0 atau 1. Output menampilkan dataframe baru dengan kolom-kolom seperti 'Prof_Engineer', 'Prof_Doctor', dll., di mana nilai 1 menunjukkan profesi yang dimiliki mahasiswa tersebut. Teknik ini memungkinkan algoritma model untuk memperlakukan setiap kategori secara independen tanpa memberikan bobot yang tidak diinginkan.

4. Binning Data Numerik Menjadi Kategori

Untuk beberapa analisis, kita perlu mengubah data numerik menjadi kategori (discretization):

```
# Melihat distribusi umur
plt.figure(figsize=(10, 6))
sns.histplot(df_student['Age'], bins=15, kde=True)
plt.title('Distribusi Umur Mahasiswa')
plt.axvline(x=df_student['Age'].mean(), color='red', linestyle='--',
label=f'Mean: {df_student["Age"].mean():.2f}')
plt.legend()
plt.show()

# Membuat kategori umur dengan pd.cut
age_bins = [0, 20, 25, 30, 100]
age_labels = ['Teen', 'Young Adult', 'Adult', 'Senior']
df_student['Age_Category'] = pd.cut(df_student['Age'], bins=age_bins,
labels=age_labels)

# Melihat distribusi kategori umur
plt.figure(figsize=(10, 6))
sns.countplot(x='Age_Category', data=df_student)
plt.title('Distribusi Kategori Umur')
plt.xticks(rotation=45)
plt.show()

# Melihat hubungan kategori umur dengan tingkat depresi
plt.figure(figsize=(12, 6))
sns.boxplot(x='Age_Category', y='Depression', data=df_student)
plt.title('Tingkat Depresi Berdasarkan Kategori Umur')
plt.show()
```

Output: Kode ini membagi nilai umur yang kontinu menjadi kategori diskrit ('Teen', 'Young Adult', 'Adult', 'Senior'). Output menampilkan:

1. Histogram distribusi umur mahasiswa
2. Diagram batang menunjukkan jumlah mahasiswa dalam setiap kategori umur

3. Boxplot yang membandingkan tingkat depresi antar kategori umur

Dengan mengkategorikan data numerik, kita dapat lebih mudah menginterpretasikan pola dan hubungan dalam data, terutama untuk komunikasi hasil analisis.

5. Kuantil-Based Binning dengan qcut

Membagi data menjadi kuantil seringkali lebih representatif daripada binning dengan interval tetap:

```
# Melihat distribusi CGPA
plt.figure(figsize=(10, 6))
sns.histplot(df_student['CGPA'], bins=15, kde=True)
plt.title('Distribusi CGPA Mahasiswa')
plt.show()

# Membuat kategori CGPA berdasarkan kuartil dengan pd.qcut
df_student['CGPA_Quartile'] = pd.qcut(df_student['CGPA'], q=4, labels=['Q1',
'Q2', 'Q3', 'Q4'])

# Melihat batas kuartil
quartile_limits = pd.qcut(df_student['CGPA'], q=4, retbins=True)[1]
print("Batas kuartil CGPA:", quartile_limits)

# Analisis tingkat depresi berdasarkan kuartil CGPA
plt.figure(figsize=(12, 6))
sns.boxplot(x='CGPA_Quartile', y='Depression', data=df_student)
plt.title('Tingkat Depresi Berdasarkan Kuartil CGPA')
plt.xlabel('Kuartil CGPA (Q1: Terendah, Q4: Tertinggi)')
plt.show()

# Persentase depresi tinggi (> 0.7) berdasarkan kuartil CGPA
high_depression = df_student['Depression'] > 0.7
depression_by_quartile =
df_student.groupby('CGPA_Quartile')[high_depression].mean() * 100
print("Persentase depresi tinggi per kuartil CGPA:")
print(depression_by_quartile)
```

Output: Kode ini membagi nilai CGPA menjadi 4 kuartil. Output menunjukkan:

1. Histogram distribusi CGPA
2. Nilai batas untuk setiap kuartil
3. Boxplot perbandingan tingkat depresi antar kuartil CGPA
4. Tabel persentase mahasiswa dengan tingkat depresi tinggi di setiap kuartil

Metode qcut ini membagi data menjadi kelompok dengan jumlah anggota yang relatif seimbang, sehingga lebih representatif untuk analisis perbandingan.

6. Pelabelan untuk Analisis Multi-dimensi

Terkadang kita perlu membuat label berdasarkan kombinasi beberapa variabel:

```
# Membuat label berdasarkan kombinasi tekanan akademik dan pekerjaan
df_student['Pressure_Level'] = 'Medium'  # Default value

# High pressure: Academic Pressure > 0.7 AND Work Pressure > 0.7
high_pressure_mask = (df_student['Academic Pressure'] > 0.7) &
(df_student['Work Pressure'] > 0.7)
df_student.loc[high_pressure_mask, 'Pressure_Level'] = 'High'

# Low pressure: Academic Pressure < 0.3 AND Work Pressure < 0.3
low_pressure_mask = (df_student['Academic Pressure'] < 0.3) &
(df_student['Work Pressure'] < 0.3)
df_student.loc[low_pressure_mask, 'Pressure_Level'] = 'Low'

# Visualisasi tingkat depresi berdasarkan level tekanan
plt.figure(figsize=(12, 6))
sns.boxplot(x='Pressure_Level', y='Depression', data=df_student,
order=['Low', 'Medium', 'High'])
plt.title('Tingkat Depresi Berdasarkan Level Tekanan')
plt.show()

# Correlation matrix
columns_of_interest = ['Academic Pressure', 'Work Pressure', 'Depression',
'CGPA']
correlation = df_student[columns_of_interest].corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Korelasi antar Variabel')
plt.tight_layout()
plt.show()
```

Output: Kode ini membuat label baru 'Pressure_Level' berdasarkan kombinasi tekanan akademik dan pekerjaan. Output menampilkan:

1. Boxplot perbandingan tingkat depresi berdasarkan level tekanan (Low, Medium, High)
2. Heatmap korelasi antar variabel yang memperlihatkan hubungan linear antara tekanan akademik, tekanan pekerjaan, depresi, dan CGPA

Dengan membuat label multi-dimensi, kita dapat mengungkap pola yang mungkin tidak terlihat ketika menganalisis variabel secara terpisah.

7. Validasi dan Evaluasi Hasil Pelabelan

Penting untuk mengevaluasi hasil pelabelan kita untuk memastikan kualitasnya: