Data Processing Utility Library - Usage Guide

Table of Contents

1. Installation
2. Basic Usage
3. Data Loading & Inspection
4. Data Cleaning
5. Feature Engineering
6. Model Prediction
7. Saving Results
8. Complete Example

Installation

bash

Copy

Download

pip install numpy pandas scikit-learn joblib streamlit

Basic Usage

python

Copy

Download

from data\_utils import load\_data, check\_data\_for\_preprocessing

*# Load your dataset*

df = load\_data('data.csv')

*# Get preprocessing recommendations*

check\_data\_for\_preprocessing(df, verbose=True)

Data Loading & Inspection

python

Copy

Download

from data\_utils import load\_data, DataFrameStatistics

*# Load data with automatic column name cleaning*

df = load\_data('sales\_data.csv')

*# Get comprehensive statistics*

stats = DataFrameStatistics(df)

stats.statistics()

*# Alternative: Get report as text file*

report = stats.generate\_report\_lines()

with open('data\_report.txt', 'w') as f:

f.write('\n'.join(report))

Data Cleaning

python

Copy

Download

from data\_utils import (handle\_missing\_values,

drop\_columns,

drop\_duplicates,

identify\_outliers)

*# Handle missing values (multiple strategies available)*

df = handle\_missing\_values(df, strategy='fill', fill\_value=0)

*# Drop unnecessary columns*

df = drop\_columns(df, ['customer\_id', 'transaction\_id'])

*# Remove duplicate rows*

df = drop\_duplicates(df)

*# Identify outliers*

outliers\_report = identify\_outliers(df)

print(outliers\_report)

Feature Engineering

python

Copy

Download

from data\_utils import encode\_column, encode\_by\_ranges

*# One-hot encode categorical column*

df = encode\_column(df, 'product\_category', encoding\_type='onehot')

*# Label encode another column*

df = encode\_column(df, 'region', encoding\_type='label')

*# Bin numeric values*

df = encode\_by\_ranges(

df,

column='customer\_age',

new\_column='age\_group',

bins=[0, 18, 35, 50, 65, 100],

labels=['0-17', '18-34', '35-49', '50-64', '65+']

)

Model Prediction

python

Copy

Download

from data\_utils import make\_prediction

*# Sample input matching your model's features*

user\_input = {

'feature1': 42.5,

'feature2': 3.7,

'feature3': 10.2

}

*# Make prediction*

prediction = make\_prediction(

user\_input=user\_input,

ModelPath='trained\_model.pkl',

ScalerPath='scaler.pkl'

)

print(f"Predicted value: {prediction[0]:.2f}")

Saving Results

python

Copy

Download

from data\_utils import save\_to\_csv, write\_to\_text\_file

*# Save processed DataFrame*

save\_to\_csv(df, 'processed\_data.csv')

*# Save statistical report*

stats = DataFrameStatistics(df)

report = stats.generate\_report\_lines()

write\_to\_text\_file(report, 'data\_analysis\_report.txt')

Complete Example

Here's an end-to-end example using the library:

python

Copy

Download

from data\_utils import (load\_data,

check\_data\_for\_preprocessing,

handle\_missing\_values,

drop\_columns,

encode\_column,

DataFrameStatistics,

save\_to\_csv)

*# 1. Load data*

print("Loading data...")

df = load\_data('customer\_data.csv')

*# 2. Data quality check*

print("\nData quality report:")

check\_data\_for\_preprocessing(df, verbose=True)

*# 3. Handle missing values*

print("\nHandling missing values...")

df = handle\_missing\_values(df, strategy='fill', fill\_value=0)

*# 4. Remove unnecessary columns*

print("\nDropping columns...")

df = drop\_columns(df, ['user\_id', 'timestamp'])

*# 5. Encode categorical features*

print("\nEncoding categorical features...")

df = encode\_column(df, 'gender', encoding\_type='onehot')

df = encode\_column(df, 'country', encoding\_type='label')

*# 6. Final statistics*

print("\nFinal statistics:")

stats = DataFrameStatistics(df)

stats.statistics()

*# 7. Save processed data*

print("\nSaving results...")

save\_to\_csv(df, 'processed\_customer\_data.csv')

print("Processing complete!")

This example shows a typical workflow:

1. Load raw data
2. Analyze data quality
3. Handle missing values
4. Remove unnecessary columns
5. Encode categorical features
6. Verify final data quality
7. Save processed data

The library is designed to make each step clear and reproducible while providing detailed feedback about the transformations being applied to your data.