UCI Adult Income Dataset - Exploratory and Descriptive Analysis

Chance UWUMUKIZA

2025-06-25

In this notebook, we focus on **data preparation**, **cleaning**, and **preprocessing** for the **UCI Adult Income Dataset**, a popular dataset often used for classification tasks predicting whether an individual earns more or less than $50,000 annually based on demographic and work-related attributes.

Good data preprocessing is crucial for reliable and interpretable results in machine learning and analytics workflows. Here, we address common data issues such as **missing values, duplicates, and inconsistent categorical labels** while creating derived features to improve downstream analysis.

We start by importing essential Python libraries for data handling and manipulation.

- `pandas` for structured data operations.
- `numpy` for numerical operations.
- `os` for interacting with the operating system and directory structures.

```
# Import libraries
import pandas as pd
import numpy as np
import os
```

Define and Create Directory Paths

To ensure reproducibility andorganized storage, we programmatically create directories for:

- **raw data**
- **processed data**
- **results**

- **documentation**

These directories will store intermediate and final outputs for reproducibility.

Define and Create Paths

```
# Get working directory
current_dir = os.getcwd()

# Go one directory up to the root directory
project_root_dir = os.path.dirname(current_dir)

# define paths to the data files
data_dir = os.path.join(project_root_dir, 'data')
raw_dir = os.path.join(data_dir, 'raw')
processed_dir = os.path.join(data_dir, 'processed')

# define paths to results folder
results_dir = os.path.join(project_root_dir, 'results')

# define paths to docs folder
docs_dir = os.path.join(project_root_dir, 'docs')

# create directories if they do not exist
os.makedirs(raw_dir, exist_ok = True)
os.makedirs(processed_dir, exist_ok = True)
os.makedirs(results_dir, exist_ok = True)
os.makedirs(docs_dir, exist_ok = True)
```

Read in the data

We load the **Adult Income dataset** as a CSV file.

Key considerations here are:

- We treat `?` as missing values (`na_values = '?'`).
- We use `skipinitialspace = True` to remove extra spaces after delimeters which is common in text-based datasets.

After loading, we inspect the first few rows.

```
adult_data_filename = os.path.join(raw_dir, "adult.csv")
adult_df = pd.read_csv(adult_data_filename, header = None, na_values = '?', skipinitialspace
adult_df.head(10)
```

```
C:\Users\USER\anaconda3\lib\site-packages\IPython\core\formatters.py:342: FutureWarning:

In future versions `DataFrame.to_latex` is expected to utilise the base implementation of `S
```

	0	1	2	3	4	5	6	7
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-fam
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-fam
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife
5	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife
6	49	Private	160187	9th	5	Married-spouse-absent	Other-service	Not-in-fam
7	52	Self-emp-not-inc	209642	HS-grad	9	Married-civ-spouse	Exec-managerial	Husband
8	31	Private	45781	Masters	14	Never-married	Prof-specialty	Not-in-fam
9	42	Private	159449	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband

We also inspect the dataset's shape. We see that the data has *32,561* rows and *15* columns.

```
adult_df.shape
```

```
(32561, 15)
```

In addition, we check the data types using `.info`.

```
adult_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   0       32561 non-null  int64
 1   1       30725 non-null  object
 2   2       32561 non-null  int64
 3   3       32561 non-null  object
```

```
 4   4       32561 non-null  int64
 5   5       32561 non-null  object
 6   6       30718 non-null  object
 7   7       32561 non-null  object
 8   8       32561 non-null  object
 9   9       32561 non-null  object
 10  10      32561 non-null  int64
 11  11      32561 non-null  int64
 12  12      32561 non-null  int64
 13  13      31978 non-null  object
 14  14      32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

Data Cleaning

1. Assign proper column names to the columns

One of the most stricking things from the above inspection is that the dataset lacks explicit column headers. We manually assign descriptive meaningful column names based on the description of the dataset. This is critical for readability and interpretability in the subsequent steps.

```
adult_df.columns = ["age", "workclass", "fnlwgt", "education", "education_num", "marital_sta
```

We inspect again to see whether they are properly assigned.

```
adult_df.head(10)
```

```
C:\Users\USER\anaconda3\lib\site-packages\IPython\core\formatters.py:342: FutureWarning:

In future versions `DataFrame.to_latex` is expected to utilise the base implementation of `S
```

	age	workclass	fnlwgt	education	education_num	marital_status	occupation
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaner
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaner
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty
5	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial
6	49	Private	160187	9th	5	Married-spouse-absent	Other-service
7	52	Self-emp-not-inc	209642	HS-grad	9	Married-civ-spouse	Exec-managerial
8	31	Private	45781	Masters	14	Never-married	Prof-specialty
9	42	Private	159449	Bachelors	13	Married-civ-spouse	Exec-managerial

2. Understanding the dataset

Before proceeding with the cleaning, we would like to understanding the variables deeply. This would help guide the cleaning process. The subsequent tables detail the types, meaning and values or ranges of the variables in the dataset.

Table 1: Summary table of the variables in the dataset

Variable	Type	Description	Values / Range (excluding nan)
age	Numeric	Age in years	17 – 90
fnlwgt	Numeric	Final sampling weight	~12,285 – 1,484,705
education_num	Numeric	Education level in years	1 – 16
capital_gain	Numeric	Capital gain amounts (Profit from selling assets above purchase price within the survey year (in USD))	0 – 99,999
capital_loss	Numeric	Capital loss amounts (Loss from selling assets below purchase price within the survey year (in USD))	0 – 4,356
hours_per_week	Numeric	Weekly work hours	1 – 99
workclass	Categorical	Type of employment	8 categories
education	Categorical	Highest level of education achieved	16 categories
marital_status	Categorical	Marital status	7 categories
occupation	Categorical	Type of job	14 categories
relationship	Categorical	Relationship within household	6 categories
race	Categorical	Ethnic/racial group	5 categories
sex	Categorical	Gender	2 categories
native_country	Categorical	Country of origin	41 categories

Variable	Type	Description	Values / Range (excluding nan)
income	Categorical	Income category (target variable)	2 categories: <=50K, >50K

Table 2: Categorical Variables Table | Variable | Unique Value | Description | |:———— ——-|:——————|:————————————————————————-| | workclass | Private | Works for a private, for-profit company | | | Self-emp-not-inc | Self-employed without incorporated business status | | | Self-emp-inc | Self-employed with an incorporated business | | | Federal-gov | Employed by the federal government | | | State-gov | Employed by a state government | | | Local-gov | Employed by a local government | | | Without-pay | Works without receiving pay (e.g. unpaid family worker) | | | Never-worked | Has never worked in their lifetime | | education | Bachelors | Bachelor's degree | | | Some-college | Some college courses completed, no degree | | | 11th | 11th grade completed | | | HS-grad | High school graduate | | | Prof-school | Professional school (e.g. law, medicine) | | | Assoc-acdm | Associate degree (academic) | | | Assoc-voc | Associate degree (vocational) | | | 9th | 9th grade completed | | | 7th-8th | 7th or 8th grade completed | | | 12th | 12th grade, no diploma | | | Masters | Master's degree | | | 1st-4th | 1st to 4th grade completed | | | 10th | 10th grade completed | | | Doctorate | Doctoral degree | | | 5th-6th | 5th or 6th grade completed | | | Preschool | Preschool education | | marital-status | Married-civ-spouse | Married, living with spouse | | | Divorced | Divorced legally | | | Never-married | Never married | | | Separated | Separated legally but not divorced | | | Widowed | Spouse deceased | | | Married-spouse-absent| Married, spouse not present (e.g. estrangement) | | | Married-AF-spouse | Married to a spouse who is a member of the Armed Forces | | occupation | Tech-support | Technical support jobs | | | Craft-repair | Skilled manual trade and repair jobs | | | Other-service | Services not classified elsewhere | | | Sales | Sales-related jobs | | | Exec-managerial | Executive and managerial roles | | | Prof-specialty | Professional specialty occupations (e.g. scientist, lawyer) | | | Handlers-cleaners | Manual labor jobs involving cleaning, handling objects | | | Machine-op-inspct | Machine operators, inspectors | | | Adm-clerical | Administrative and clerical jobs | | | Farming-fishing | Agriculture, farming, fishing occupations | | | Transport-moving | Transport and moving equipment operators | | | Priv-house-serv | Private household service jobs | | | Protective-serv | Protective service jobs (e.g. security, law enforcement) | | | Armed-Forces | Military service | | relationship | Wife | Female spouse | | | Own-child | Biological or adopted child | | | Husband | Male spouse | | | Not-in-family | Not part of a family unit (e.g. living alone) | | | Other-relative | Other relative in household | | | Unmarried | Single person, not married | | race | White | White | | | Asian-Pac-Islander | Asian or Pacific Islander | | | Amer-Indian-Eskimo | American Indian or Eskimo | | | Other | Other race not listed | | | Black | Black | | sex | Female | Female | | | Male | Male | | native-country | United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland,

Thailand, Yugoslavia, El-Salvador, Trinidad-Tobago, Peru, Hong, Holland-Netherlands | | | income | <=50K | Income less than or equal to USD 50,000 | | | >50K | Income greater than USD 50,000 |

```
np.unique(adult_df.age.to_list())
```

```
array([17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
       34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
       51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
       68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,
       85, 86, 87, 88, 90])
```

```
np.unique(adult_df.workclass.to_list())
```

```
array(['Federal-gov', 'Local-gov', 'Never-worked', 'Private',
       'Self-emp-inc', 'Self-emp-not-inc', 'State-gov', 'Without-pay',
       'nan'], dtype='<U32')
```

```
np.unique(adult_df.fnlwgt.to_list())
```

```
array([  12285,   13769,   14878, ..., 1366120, 1455435, 1484705])
```

```
np.unique(adult_df.education_num.to_list())
```

```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16])
```

```
adult_df.columns
```

```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education_num',
       'marital_status', 'occupation', 'relationship', 'race', 'sex',
       'capital_gain', 'capital_loss', 'hours_per_week', 'native_country',
       'income'],
      dtype='object')
```

```
np.unique(adult_df.marital_status.to_list())
```

```
array(['Divorced', 'Married-AF-spouse', 'Married-civ-spouse',
       'Married-spouse-absent', 'Never-married', 'Separated', 'Widowed'],
      dtype='<U21')
```

```python
np.unique(adult_df.occupation.to_list())
```

```
array(['Adm-clerical', 'Armed-Forces', 'Craft-repair', 'Exec-managerial',
       'Farming-fishing', 'Handlers-cleaners', 'Machine-op-inspct',
       'Other-service', 'Priv-house-serv', 'Prof-specialty',
       'Protective-serv', 'Sales', 'Tech-support', 'Transport-moving',
       'nan'], dtype='<U32')
```

```python
np.unique(adult_df.relationship.to_list())
```

```
array(['Husband', 'Not-in-family', 'Other-relative', 'Own-child',
       'Unmarried', 'Wife'], dtype='<U14')
```

```python
np.unique(adult_df.sex.to_list())
```

```
array(['Female', 'Male'], dtype='<U6')
```

```python
np.unique(adult_df.capital_gain.to_list())
```

```
array([    0,   114,   401,   594,   914,   991,  1055,  1086,  1111,
        1151,  1173,  1409,  1424,  1455,  1471,  1506,  1639,  1797,
        1831,  1848,  2009,  2036,  2050,  2062,  2105,  2174,  2176,
        2202,  2228,  2290,  2329,  2346,  2354,  2387,  2407,  2414,
        2463,  2538,  2580,  2597,  2635,  2653,  2829,  2885,  2907,
        2936,  2961,  2964,  2977,  2993,  3103,  3137,  3273,  3325,
        3411,  3418,  3432,  3456,  3464,  3471,  3674,  3781,  3818,
        3887,  3908,  3942,  4064,  4101,  4386,  4416,  4508,  4650,
        4687,  4787,  4865,  4931,  4934,  5013,  5060,  5178,  5455,
        5556,  5721,  6097,  6360,  6418,  6497,  6514,  6723,  6767,
        6849,  7298,  7430,  7443,  7688,  7896,  7978,  8614,  9386,
        9562, 10520, 10566, 10605, 11678, 13550, 14084, 14344, 15020,
       15024, 15831, 18481, 20051, 22040, 25124, 25236, 27828, 34095,
       41310, 99999])
```

```python
np.unique(adult_df.capital_loss.to_list())
```

```
array([   0,  155,  213,  323,  419,  625,  653,  810,  880,  974, 1092,
       1138, 1258, 1340, 1380, 1408, 1411, 1485, 1504, 1539, 1564, 1573,
       1579, 1590, 1594, 1602, 1617, 1628, 1648, 1651, 1668, 1669, 1672,
       1719, 1721, 1726, 1735, 1740, 1741, 1755, 1762, 1816, 1825, 1844,
       1848, 1876, 1887, 1902, 1944, 1974, 1977, 1980, 2001, 2002, 2042,
       2051, 2057, 2080, 2129, 2149, 2163, 2174, 2179, 2201, 2205, 2206,
       2231, 2238, 2246, 2258, 2267, 2282, 2339, 2352, 2377, 2392, 2415,
       2444, 2457, 2467, 2472, 2489, 2547, 2559, 2603, 2754, 2824, 3004,
       3683, 3770, 3900, 4356])
```

```
np.unique(adult_df.hours_per_week.to_list())
```

```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
       35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51,
       52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68,
       70, 72, 73, 74, 75, 76, 77, 78, 80, 81, 82, 84, 85, 86, 87, 88, 89,
       90, 91, 92, 94, 95, 96, 97, 98, 99])
```

```
np.unique(adult_df.native_country.to_list())
```

```
array(['Cambodia', 'Canada', 'China', 'Columbia', 'Cuba',
       'Dominican-Republic', 'Ecuador', 'El-Salvador', 'England',
       'France', 'Germany', 'Greece', 'Guatemala', 'Haiti',
       'Holand-Netherlands', 'Honduras', 'Hong', 'Hungary', 'India',
       'Iran', 'Ireland', 'Italy', 'Jamaica', 'Japan', 'Laos', 'Mexico',
       'Nicaragua', 'Outlying-US(Guam-USVI-etc)', 'Peru', 'Philippines',
       'Poland', 'Portugal', 'Puerto-Rico', 'Scotland', 'South', 'Taiwan',
       'Thailand', 'Trinadad&Tobago', 'United-States', 'Vietnam',
       'Yugoslavia', 'nan'], dtype='<U32')
```

```
np.unique(adult_df.income.to_list())
```

```
array(['<=50K', '>50K'], dtype='<U5')
```

3. Deal with missing values