# UCI Adult Income Dataset - Exploratory and Descriptive Analysis

In this notebook, we carry out an in-depth exploratory and descriptive analysis of the UCI Adult Income Dataset, a widely used dataset for income prediction tasks based on individual demographic and employment attributes.

This phase of analysis is essential for uncovering patterns, detecting potential biases, and gaining intuition about the dataset's structure before applying any modelling procedures. We examine the distribution of key numerical and categorical variables, investigate relationships between demographic features and income levels, and use visualizations to summarize insights. Particular focus is placed on income disparities across **age groups**, **geographical regions**, **races**, and **education-occupation combinations**, helping lay a solid foundation for downstream modeling and policy-relevant interpretation.

We begin our analysis by importing the core Python libraries required for **data handling**, **numerical computation**, **visualization**, and **directory management**:

- `pandas`: Enables efficient manipulation, filtering, and aggregation of structured tabular data, forming the backbone of our analysis pipeline.
- `numpy`: Provides support for fast numerical operations, array-based computation, and statistical routines.
- `os`: Facilitates interaction with the file system, allowing us to construct flexible and portable directory paths for data and output management.
- `plotly.express`: A high-level graphing library that enables the creation of interactive, publication-quality visualizations, which we use extensively to uncover patterns and present insights throughout the notebook.

```
# import libraries
import pandas as pd
import numpy as np
import os
import plotly.express as px
```

### Define and Create Directory Paths

To ensure reproducibility andorganized storage, we programmatically create directories if they don't already exist for:

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

These directories will store intermediate and final outputs for reproducibility.

```python
# Get working directory
current_dir = os.getcwd()
# go to directory up to the root directory
project_root_dir = os.path.dirname(current_dir)
# Define paths to the data folders
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 the data folders
result_dir = os.path.join(project_root_dir, 'result')
# Define paths to docs folderabs
docs_dir = os.path.join(project_root_dir, 'docs')

# create directories if the do not eFile exists
os.makedirs(raw_dir, exist_ok = True)
os.makedirs(processed_dir, exist_ok = True)
os.makedirs(result_dir, exist_ok = True)
os.makedirs(docs_dir, exist_ok = True)
```

### Loading the Cleaned Dataset

We load the cleaned version of the UCI Adult Income Dataset from the processed data directory into a Pandas DataFrame. The `head(10)` function shows the first ten records, giving a glimpse into the data columns such as `age`, `workclass`, `education_num`, etc.

```python
adult_data_filename = os.path.join(processed_dir, "adult_cleaned.csv")
adult_df = pd.read_csv(adult_data_filename)
adult_df.head(10)
```

| | age | workclass | fnlwgt | education_num | marital_status | relationship | race | sex | c |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 39 | government | 77516 | 13 | single | single | white | male | 2 |
| 1 | 50 | self-employed | 83311 | 13 | married | male spouse | white | male | 0 |
| 2 | 38 | private | 215646 | 9 | divorced or separated | single | white | male | 0 |
| 3 | 53 | private | 234721 | 7 | married | male spouse | black | male | 0 |
| 4 | 28 | private | 338409 | 13 | married | female spouse | black | female | 0 |
| 5 | 37 | private | 284582 | 14 | married | female spouse | white | female | 0 |
| 6 | 49 | private | 160187 | 5 | divorced or separated | single | black | female | 0 |
| 7 | 52 | self-employed | 209642 | 9 | married | male spouse | white | male | 0 |
| 8 | 31 | private | 45781 | 14 | single | single | white | female | 1 |
| 9 | 42 | private | 159449 | 13 | married | male spouse | white | male | 5 |

### Dataset Dimensions and Data Types

Here, we examine the structure of the dataset:

- There are *32,513* entries and *16* variables.
- The dataset includes both **numerical** (e.g., `age`, `hours_per_week`) and **categorical** variables (e.g., `sex`, `education_level`).

Understanding data types and null entries is essential before proceeding with analysis.

```
adult_df.shape
```

```
(32533, 16)
```

```
adult_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32533 entries, 0 to 32532
Data columns (total 16 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   age                 32533 non-null  int64
 1   workclass           32533 non-null  object
 2   fnlwgt              32533 non-null  int64
 3   education_num       32533 non-null  int64
 4   marital_status      32533 non-null  object
 5   relationship        32533 non-null  object
 6   race                32533 non-null  object
```

```
 7   sex                 32533 non-null  object
 8   capital_gain        32533 non-null  int64
 9   capital_loss        32533 non-null  int64
 10  hours_per_week      32533 non-null  int64
 11  income              32533 non-null  object
 12  education_level     32533 non-null  object
 13  occupation-grouped  32533 non-null  object
 14  native_region       32533 non-null  object
 15  age_group           32533 non-null  object
dtypes: int64(6), object(10)
memory usage: 4.0+ MB
```

### summary statistics

#### Numerical variables

```python
adult_df.describe()       #  sumaries the numerical data
```

| | age | fnlwgt | education_num | capital_gain | capital_loss | hours_per_week |
|---|---|---|---|---|---|---|
| count | 32533.000000 | 3.253300e+04 | 32533.000000 | 32533.000000 | 32533.000000 | 32533.000000 |
| mean | 38.587557 | 1.897849e+05 | 10.081640 | 1078.576338 | 87.378969 | 40.441306 |
| std | 13.637609 | 1.055601e+05 | 2.571689 | 7388.401928 | 403.125450 | 12.346494 |
| min | 17.000000 | 1.228500e+04 | 1.000000 | 0.000000 | 0.000000 | 1.000000 |
| 25% | 28.000000 | 1.178330e+05 | 9.000000 | 0.000000 | 0.000000 | 40.000000 |
| 50% | 37.000000 | 1.783560e+05 | 10.000000 | 0.000000 | 0.000000 | 40.000000 |
| 75% | 48.000000 | 2.369930e+05 | 12.000000 | 0.000000 | 0.000000 | 45.000000 |
| max | 90.000000 | 1.484705e+06 | 16.000000 | 99999.000000 | 4356.000000 | 99.000000 |

This summary provides a snapshot of key distribution characteristics. We see that:

- Age ranges from 17 to 90, with a mean of 38.6 years. It is slightly right-skewed (positively skewed). While the average age is approximately 38.6 years, an examination of the percentiles reveals that the majority of individuals are clustered in the younger to middle-age range, with fewer observations in the older age brackets. This skewed age distribution might suggest labor force participation is concentrated in specific age groups, which could reflect broader demographic or economic realities.
- Capital gains/losses are highly skewed, with most values at 0 (the 75th percentile is 0). This indicates that a small number of individuals report very large gains or losses,

especially evident in the capital gain variable which reaches up to $99,999. These variables act as proxies for wealth-related income that goes beyond regular wages or salaries. Individuals with non-zero values for capital gains or losses often represent a distinct socioeconomic subset of the population — typically more financially literate, or with access to investment assets. The stark inequality in their distributions mirrors real-world disparities in asset ownership and investment returns.

- The dataset has individuals working anywhere from 1 to 99 hours per week, with a median of 40. This aligns with the standard full-time work week in many countries (8 hours per day for 5 working days). The mean is slightly above that at 40.4 hours, suggesting a mild right skew, with a small subset of individuals working significantly longer hours. The mode is also 40, further reinforcing the prevalence of full-time work. A non-trivial number of individuals report working very few hours, possibly due to part-time work, unemployment, or semi-retirement. On the other extreme, some report working more than 45 hours per week, which may indicate multiple jobs, weekend-work, self-employment, or informal labor, and could reflect socioeconomic necessity.

### Categorical variables

```
adult_df.describe(include='object')
```

| | workclass | marital_status | relationship | race | sex | income | education_level | occupatio |
|---|---|---|---|---|---|---|---|---|
| count | 32533 | 32533 | 32533 | 32533 | 32533 | 32533 | 32533 | 32533 |
| unique | 6 | 4 | 5 | 5 | 2 | 2 | 7 | 9 |
| top | private | married | male spouse | white | male | <=50k | secondary graduate | white colle |
| freq | 22670 | 14993 | 13187 | 27791 | 21774 | 24694 | 10494 | 11480 |

```
adult_df['workclass'].value_counts(normalize=True)
```

```
workclass
private          0.696831
government       0.133710
self-employed    0.112378
unknown          0.056435
voluntary        0.000430
unemployed       0.000215
Name: proportion, dtype: float64
```

```python
adult_df['marital_status'].value_counts(normalize=True)
```

```
marital_status
married                  0.460855
single                   0.327760
divorced or separated    0.180863
widowed                  0.030523
Name: proportion, dtype: float64
```

```python
adult_df['relationship'].value_counts(normalize=True)
```

```
relationship
male spouse          0.405342
single               0.360680
child                0.155627
female spouse        0.048197
extended relative    0.030154
Name: proportion, dtype: float64
```

```python
adult_df['race'].value_counts(normalize=True)
```

```
race
white                        0.854240
black                        0.095964
asian or pacific islander    0.031906
american indian or eskimo    0.009560
other                        0.008330
Name: proportion, dtype: float64
```

```python
adult_df['sex'].value_counts(normalize=True)
```

```
sex
male      0.66929
female    0.33071
Name: proportion, dtype: float64
```

```python
adult_df['education_level'].value_counts(normalize=True)
```

```
education_level
secondary graduate    0.322565
tertiary              0.247810
unemployed            0.223773
secondary             0.093905
associate             0.075277
primary               0.035134
preschool             0.001537
Name: proportion, dtype: float64
```

```python
adult_df['occupation-grouped'].value_counts(normalize=True)
```

```
occupation-grouped
white collor    0.352872
blue collar     0.156334
white color     0.127132
service         0.125626
blue collor     0.091169
whitecollar     0.061476
unknown         0.056650
white collar    0.028463
military        0.000277
Name: proportion, dtype: float64
```

```python
adult_df['native_region'].value_counts(normalize=True)
```

```
native_region
north america      0.923278
asia               0.020625
other              0.017890
central america    0.016107
europe             0.016015
south america      0.006086
Name: proportion, dtype: float64
```

```python
adult_df['age_group'].value_counts(normalize=True)
```

```
age_group
26-35    0.261581
36-45    0.246058
```

```
46-60    0.224111
18-25    0.167553
61-75    0.064273
<18      0.029047
76+      0.007377
Name: proportion, dtype: float64
```

`workclass`

The private sector dominates, employing ~69.7% of the population. The government sector (13.4%) and self-employment (11.2%) also make up substantial portions of the workforce. A small fraction is labeled as "unknown" (5.6%), which may correspond to missing or ambiguous data entries. Tiny proportions are voluntary (0.04%) or unemployed (0.02%), possibly underreported or underrepresented in the sample.

`marital_status`

Married individuals make up the largest group (46.1%), followed by those who are single (32.8%) and divorced or separated (18.1%). Widowed individuals represent a small minority (~3.1%).

`relationship`

The majority are labeled as "male spouse" (40.5%) or "single" (36.1%). Smaller categories include children (15.6%), female spouses (4.8%), and extended relatives (3.0%). The dominance of `male spouse` reflects the dataset's gendered structure and may point to traditional family roles. The relative scarcity of "female spouse" roles suggests potential gender imbalances in how income-earning is reported within households.

`race`

The dataset is overwhelmingly composed of White individuals (~85.4%). Other racial groups include Black (9.6%), Asian or Pacific Islander (3.2%), American Indian or Eskimo (1.0%), and Other (0.8%). The racial imbalance limits the generalizability of models trained on this data. Smaller racial groups may suffer from limited statistical power, affecting fairness and performance in predictive modeling.

`sex`

Males constitute 66.9% of the dataset, with females making up the remaining 33.1%. This male-skewed distribution could be due to sampling (e.g., primary earners in households), workforce participation patterns, or reporting biases.

`education_level`

Secondary-school graduates form the largest educational group (~32%), highlighting the central role of high school completion in the labor force. Tertiary education holders — those with university or equivalent degrees — account for nearly 25% of the population, representing a substantial segment with advanced qualifications. A notable 22.4% have attended some college without necessarily earning a degree, suggesting that partial post-secondary education is common, yet may not always translate into formal certification. The remaining 20% are distributed among those with only secondary education (9.4%), associate degrees (7.5%), primary school (3.5%), and a very small group with only preschool education (0.15%). It is ecident that the education distribution is skewed toward mid- to high-level education, with relatively few individuals having only basic schooling. This reflects a dataset that largely captures working-age adults in formal labor, which may underrepresent the least-educated populations.

`occupation_grouped`

White-collar occupations are the most prevalent (~51%), followed by blue-collar, service, and unknown. Smaller categories include military, which is marginal. Essentially, slightly over half of individuals in the dataset work in professional, managerial, sales, clerical, or tech-support roles. This suggests the dataset is heavily weighted toward professional and administrative occupations. Nearly a third of the population works in manual labor or skilled trade positions (craft, transport, machine operation, farming, etc.). This indicates a significant segment engaged in physically intensive or technical labor.

`native_region`

The vast majority of individuals are from North America (~92.3%). Smaller proportions are from Central America, Asia, Europe, South America, and a generic Other category. The heavy concentration of North American individuals reflects the U.S. focus of the dataset.

`age_group`

The largest groups are 26–35 and 36–45, followed by 46–60. These three age groups represent about 73% of the dataset. Very few individuals are under 18 or above 75, consistent with the dataset's focus on the working-age population.

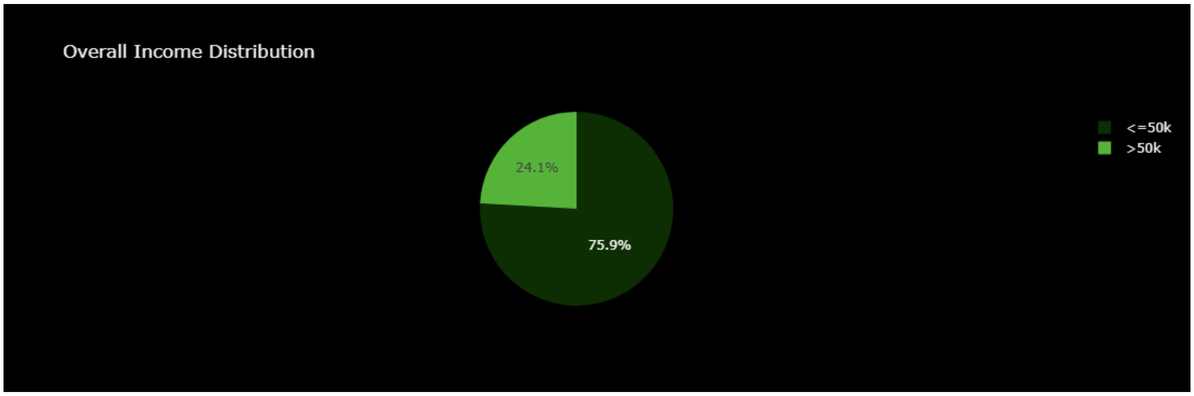
## Income Distribution

Given that `income` is the target variable, most of the analysis hereafter will be based on it. We first of all examine the income distribution in the dataset.

```
adult_df_income = adult_df.groupby('income').size().reset_index(name='total')
adult_df_income
```

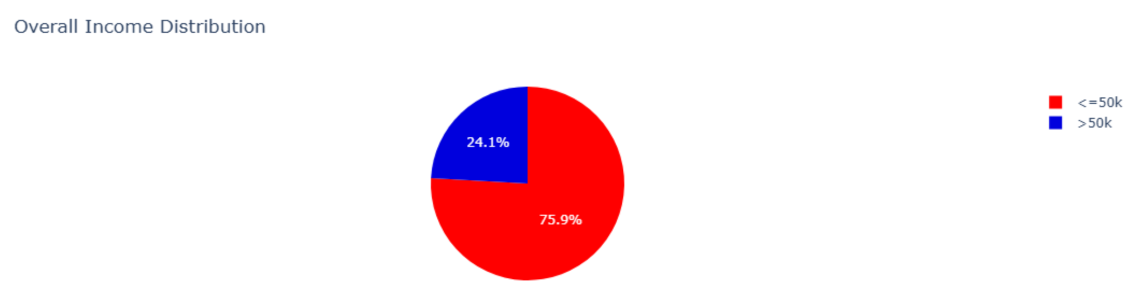
| | income | total |
|---|---|---|
| 0 | <=50k | 24694 |
| 1 | >50k | 7839 |

```
fig = px.pie(adult_df_income, names='income' , values='total', title='Overall Income Distribu
fig.update_layout(
    template='plotly_dark',  # Safe background styling
    paper_bgcolor='black'
)
fig.show()
fig.write_image(os.path.join(result_dir, 'income_distribution_pie_chart.jpg'))
fig.write_image(os.path.join(result_dir, 'income_distribution_pie_chart.png'))
fig.write_html(os.path.join(result_dir, 'income_distribution_pie_chart.html'))
```




```
WARNING Thread(Thread-57 (run)) Task(Task-944) choreographer.browser_async:browser_async.py:
```

```
fig = px.pie(adult_df_income, names='income' , values='total', title='Overall Income Distribu
fig.show()
```




This pie chart visualizes the overall income split: 76% of individuals earn 50K, while 24% earn >50K. This means that nearly 3 out of 4 individuals fall into the lower income bracket (<=50K). This shows that there is a significant imbalance.

### Income by Age group

```
adult_df_income_age= adult_df.groupby(['age_group', 'income']).size().reset_index(name='tota
adult_df_income_age
```

| | age_group | income | total_by_age |
|---|---|---|---|
| 0 | 18-25 | <=50k | 5337 |
| 1 | 18-25 | >50k | 114 |
| 2 | 26-35 | <=50k | 6919 |
| 3 | 26-35 | >50k | 1591 |
| 4 | 36-45 | <=50k | 5233 |
| 5 | 36-45 | >50k | 2772 |
| 6 | 46-60 | <=50k | 4480 |
| 7 | 46-60 | >50k | 2811 |
| 8 | 61-75 | <=50k | 1580 |
| 9 | 61-75 | >50k | 511 |
| 10 | 76+ | <=50k | 200 |
| 11 | 76+ | >50k | 40 |
| 12 | <18 | <=50k | 945 |

```
adult_df_income_age = adult_df.groupby(['age_group', 'income']).size().reset_index(name='tot
adult_df_income_age
```

| | age_group | income | total_by_age |
|---|---|---|---|
| 0 | 18-25 | <=50k | 5337 |
| 1 | 18-25 | >50k | 114 |
| 2 | 26-35 | <=50k | 6919 |
| 3 | 26-35 | >50k | 1591 |
| 4 | 36-45 | <=50k | 5233 |
| 5 | 36-45 | >50k | 2772 |
| 6 | 46-60 | <=50k | 4480 |
| 7 | 46-60 | >50k | 2811 |
| 8 | 61-75 | <=50k | 1580 |
| 9 | 61-75 | >50k | 511 |
| 10 | 76+ | <=50k | 200 |
| 11 | 76+ | >50k | 40 |
| 12 | <18 | <=50k | 945 |

```python
total_per_group = adult_df_income_age.groupby('age_group').size()
total_per_group
```

```
age_group
18-25    2
26-35    2
36-45    2
46-60    2
61-75    2
76+      2
<18      1
dtype: int64
```

```python
total_per_group = adult_df_income_age.groupby('age_group')['total_by_age'].transform('sum')
total_per_group
```

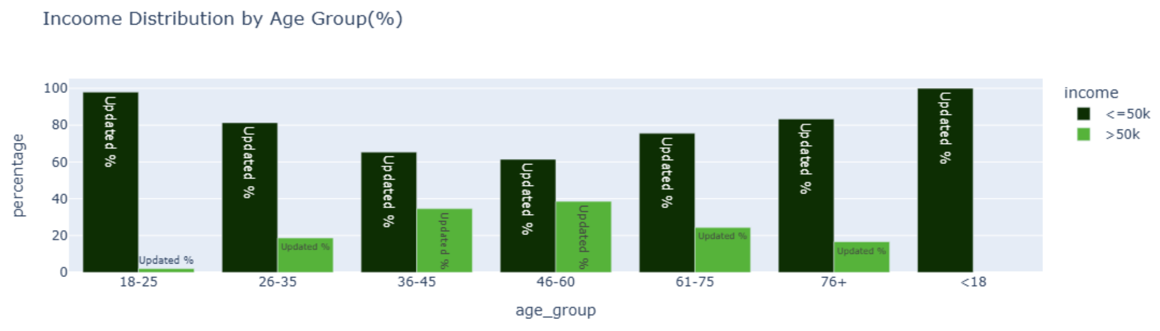
```
0     5451
1     5451
2     8510
3     8510
4     8005
5     8005
6     7291
7     7291
8     2091
9     2091
10     240
11     240
12     945
Name: total_by_age, dtype: int64
```

```python
total_per_group = adult_df_income_age.groupby('age_group')['total_by_age'].transform('sum')
adult_df_income_age['percentage'] = (adult_df_income_age['total_by_age']/total_per_group) * 
adult_df_income_age
```

| | age_group | income | total_by_age | percentage |
|---|---|---|---|---|
| 0 | 18-25 | <=50k | 5337 | 97.908641 |
| 1 | 18-25 | >50k | 114 | 2.091359 |
| 2 | 26-35 | <=50k | 6919 | 81.304348 |

| | age_group | income | total_by_age | percentage |
|---|---|---|---|---|
| 3 | 26-35 | >50k | 1591 | 18.695652 |
| 4 | 36-45 | <=50k | 5233 | 65.371643 |
| 5 | 36-45 | >50k | 2772 | 34.628357 |
| 6 | 46-60 | <=50k | 4480 | 61.445618 |
| 7 | 46-60 | >50k | 2811 | 38.554382 |
| 8 | 61-75 | <=50k | 1580 | 75.561932 |
| 9 | 61-75 | >50k | 511 | 24.438068 |
| 10 | 76+ | <=50k | 200 | 83.333333 |
| 11 | 76+ | >50k | 40 | 16.666667 |
| 12 | <18 | <=50k | 945 | 100.000000 |

```
fig = px.bar(
    adult_df_income_age,
    x = 'age_group',
    y = 'percentage',
    color = 'income',
    title='Incoome Distribution by Age Group(%)',
    barmode='group',
color_discrete_sequence=['#0d2e03', '#56b33a'],
)
fig.update_traces(texttemplate = 'Updated %')
fig.show()
fig.write_image(os.path.join(result_dir, 'income_distribution_by_agegroupchart.jpg'))
fig.write_image(os.path.join(result_dir, 'income_distribution_pie_chart.png'))
fig.write_image(os.path.join(result_dir, 'income_distribution_pie_chart.html'))
themes = ["plotly", "ploty_white", "plotly_dark", "ggplot2","seasborn", "simple_white","prese
for theme in themes:
    fig.update_layout(template=theme)
    fig.show()
```



```
WARNING Thread(Thread-65 (run)) Task(Task-1080) choreographer.browser_async:browser_async.py

RuntimeError: Couldn't close or kill browser subprocess
---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
Cell In[75], line 12
     10 fig.update_traces(texttemplate = 'Updated %')
     11 fig.show()
---> 12 fig.write_image(os.path.join(result_dir, 'income_distribution_by_agegroupchart.jpg'))
     13 fig.write_image(os.path.join(result_dir, 'income_distribution_pie_chart.png'))
     14 fig.write_image(os.path.join(result_dir, 'income_distribution_pie_chart.html'))
File ~\anaconda3\Lib\site-packages\plotly\basedatatypes.py:3911, in BaseFigure.write_image(se
   3907     if kwargs.get("engine", None):
   3908         warnings.warn(
   3909             ENGINE_PARAM_DEPRECATION_MSG, DeprecationWarning, stacklevel=2
   3910         )
-> 3911 return pio.write_image(self, *args, **kwargs)
File ~\anaconda3\Lib\site-packages\plotly\io\_kaleido.py:509, in write_image(fig, file, forma
    505 format = infer_format(path, format)
    507 # Request image
    508 # Do this first so we don't create a file if image conversion fails
--> 509 img_data = to_image(
    510     fig,
    511     format=format,
    512     scale=scale,
    513     width=width,
    514     height=height,
    515     validate=validate,
    516     engine=engine,
    517 )
    519 # Open file
    520 if path is None:
    521     # We previously failed to make sense of `file` as a pathlib object.
    522     # Attempt to write to `file` as an open file descriptor.
File ~\anaconda3\Lib\site-packages\plotly\io\_kaleido.py:373, in to_image(fig, format, width
    369 from kaleido.errors import ChromeNotFoundError
    371 try:
    372     # TODO: Refactor to make it possible to use a shared Kaleido instance here
--> 373     img_bytes = kaleido.calc_fig_sync(
    374         fig_dict,
    375         opts=dict(
    376             format=format or defaults.default_format,
    377             width=width or defaults.default_width,
```

```
    378             height=height or defaults.default_height,
    379             scale=scale or defaults.default_scale,
    380         ),
    381         topojson=defaults.topojson,
    382         kopts=(
    383             dict(
    384                 mathjax=defaults.mathjax,
    385             )
    386             if defaults.mathjax
    387             else None
    388         ),
    389     )
    390 except ChromeNotFoundError:
    391     raise RuntimeError(PLOTLY_GET_CHROME_ERROR_MSG)
File ~\anaconda3\Lib\site-packages\kaleido\__init__.py:145, in calc_fig_sync(*args, **kwargs
    143 def calc_fig_sync(*args, **kwargs):
    144     """Call `calc_fig` but blocking."""
--> 145     return _async_thread_run(calc_fig, args=args, kwargs=kwargs)
File ~\anaconda3\Lib\site-packages\kaleido\__init__.py:138, in _async_thread_run(func, args,
    136 res = q.get()
    137 if isinstance(res, BaseException):
--> 138     raise res
    139 else:
    140     return res
File ~\anaconda3\Lib\site-packages\kaleido\__init__.py:129, in _async_thread_run.<locals>.ru
    126 def run(*args, **kwargs):
    127     # func is a closure
    128     try:
--> 129         q.put(asyncio.run(func(*args, **kwargs)))
    130     except BaseException as e:  # noqa: BLE001
    131         q.put(e)
File ~\anaconda3\Lib\asyncio\runners.py:194, in run(main, debug, loop_factory)
    190     raise RuntimeError(
    191         "asyncio.run() cannot be called from a running event loop")
    193 with Runner(debug=debug, loop_factory=loop_factory) as runner:
--> 194     return runner.run(main)
File ~\anaconda3\Lib\asyncio\runners.py:118, in Runner.run(self, coro, context)
    116 self._interrupt_count = 0
    117 try:
--> 118     return self._loop.run_until_complete(task)
    119 except exceptions.CancelledError:
    120     if self._interrupt_count > 0:
File ~\anaconda3\Lib\asyncio\base_events.py:687, in BaseEventLoop.run_until_complete(self, f
```

```
    684 if not future.done():
    685     raise RuntimeError('Event loop stopped before Future completed.')
--> 687 return future.result()
File ~\anaconda3\Lib\site-packages\kaleido\__init__.py:54, in calc_fig(fig, path, opts, topo
     52 kopts = kopts or {}
     53 kopts["n"] = 1
---> 54 async with Kaleido(**kopts) as k:
     55     return await k.calc_fig(
     56         fig,
     57         path=path,
     58         opts=opts,
     59         topojson=topojson,
     60     )
File ~\anaconda3\Lib\site-packages\kaleido\kaleido.py:76, in Kaleido.__aexit__(self, exc_type
     74 await asyncio.gather(*self._background_render_tasks, return_exceptions=True)
     75 _logger.info("Exiting Kaleido")
---> 76 return await super().__aexit__(exc_type, exc_value, exc_tb)
File ~\anaconda3\Lib\site-packages\choreographer\browser_async.py:249, in Browser.__aexit__(s
    242 async def __aexit__(
    243     self,
    244     type_: type[BaseException] | None,
    245     value: BaseException | None,
    246     traceback: TracebackType | None,
    247 ) -> None:  # None instead of False is fine, eases type checking
    248     """Close the browser."""
--> 249     await self.close()
File ~\anaconda3\Lib\site-packages\kaleido\kaleido.py:69, in Kaleido.close(self)
     67         task.cancel()
     68 _logger.info("Exiting Kaleido/Choreo")
---> 69 return await super().close()
File ~\anaconda3\Lib\site-packages\choreographer\browser_async.py:228, in Browser.close(self)
    226 try:
    227     _logger.debug("Starting browser close methods.")
--> 228     await self._close()
    229     _logger.debug("Browser close methods finished.")
    230 except ProcessLookupError:
File ~\anaconda3\Lib\site-packages\choreographer\browser_async.py:216, in Browser._close(sel
    214     return
    215 else:
--> 216     raise RuntimeError("Couldn't close or kill browser subprocess")
RuntimeError: Couldn't close or kill browser subprocess
```

```
pip install -U kaleido
```

```
pip install -U plotly
```

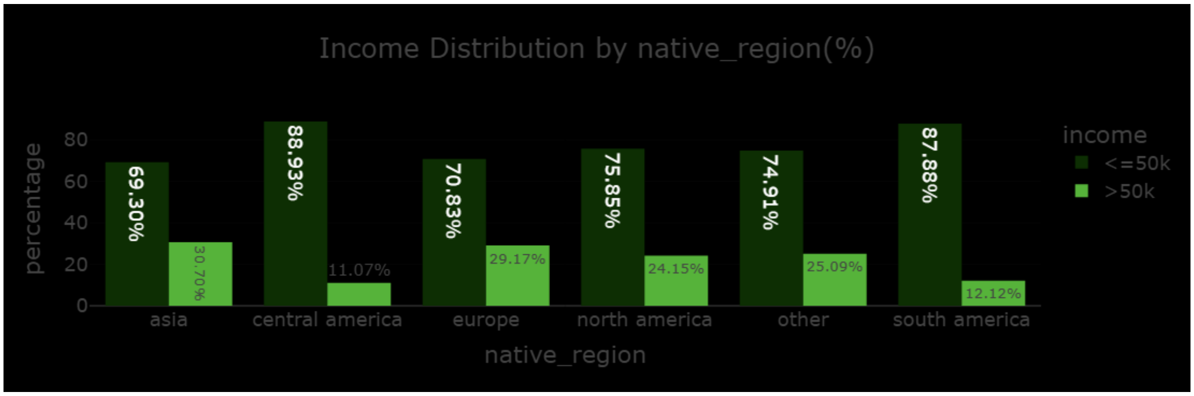
```
adult_df_income_native_region = adult_df.groupby(['native_region', 'income']).size().reset_ir
adult_df_income_native_region
```

| | native_region | income | total_income_distr |
|---|---|---|---|
| 0 | asia | <=50k | 465 |
| 1 | asia | >50k | 206 |
| 2 | central america | <=50k | 466 |
| 3 | central america | >50k | 58 |
| 4 | europe | <=50k | 369 |
| 5 | europe | >50k | 152 |
| 6 | north america | <=50k | 22784 |
| 7 | north america | >50k | 7253 |
| 8 | other | <=50k | 436 |
| 9 | other | >50k | 146 |
| 10 | south america | <=50k | 174 |
| 11 | south america | >50k | 24 |

```
total_per_region = adult_df_income_native_region.groupby('native_region')['total_income_dist
adult_df_income_native_region['percentage'] = (adult_df_income_native_region['total_income_di
adult_df_income_native_region
```

| | native_region | income | total_income_distr | percentage |
|---|---|---|---|---|
| 0 | asia | <=50k | 465 | 69.299553 |
| 1 | asia | >50k | 206 | 30.700447 |
| 2 | central america | <=50k | 466 | 88.931298 |
| 3 | central america | >50k | 58 | 11.068702 |
| 4 | europe | <=50k | 369 | 70.825336 |
| 5 | europe | >50k | 152 | 29.174664 |
| 6 | north america | <=50k | 22784 | 75.853114 |
| 7 | north america | >50k | 7253 | 24.146886 |
| 8 | other | <=50k | 436 | 74.914089 |
| 9 | other | >50k | 146 | 25.085911 |
| 10 | south america | <=50k | 174 | 87.878788 |
| 11 | south america | >50k | 24 | 12.121212 |

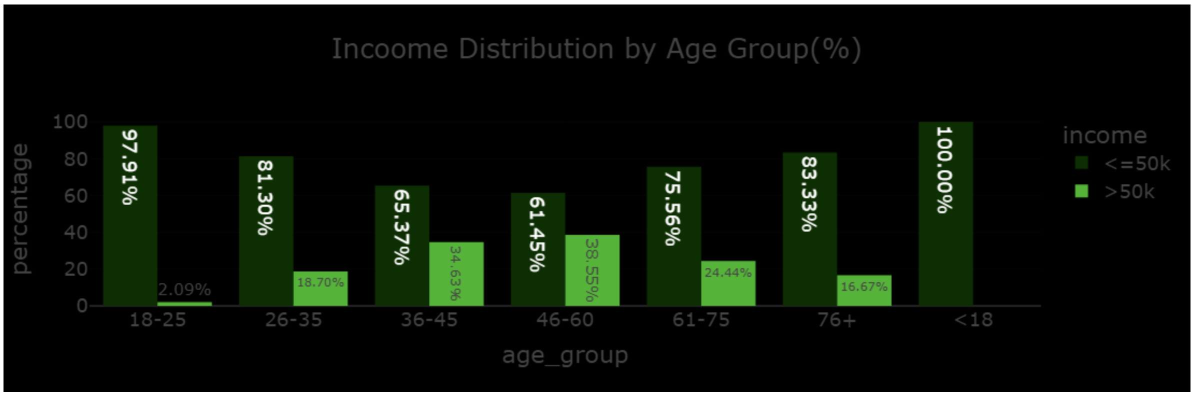
```
fig = px.bar(
    adult_df_income_native_region,
    x = 'native_region',
    y = 'percentage',
    color = 'income',
    title='Income Distribution by native_region(%)',
    barmode='group',
    color_discrete_sequence=['#0d2e03', '#56b33a'],
    text='percentage'
)
fig.update_traces(texttemplate = '%{text:.2f}%')
fig.update_layout(template='presentation', paper_bgcolor='rgb(0, 0, 0)', plot_bgcolor='rgb(0
fig.show()
fig.write_image(os.path.join(result_dir, 'income_distribution_by_nativeregion_bar_plot.jpg')
fig.write_image(os.path.join(result_dir, 'income_distribution_by_nativeregion_bar_plot.png')
fig.write_html(os.path.join(result_dir, 'income_distribution_by_nativeregion_bar_plot.html')
```




```
WARNING Thread(Thread-71 (run)) Task(Task-1199) choreographer.browser_async:browser_async.py
```

Asia (30.7%) and Europe (29.2%) have the highest proportions of high-income earners. This suggests these immigrant groups might be better integrated into high-paying professional roles, or may represent a more skilled migrant profile in the dataset. Central America (11.1%) and South America (12.1%) have the lowest proportions of >50K earners. With 24.2% of North Americans earning >50K, this serves as a middle-ground baseline. Interestingly, both Asian and European groups outperform the native-born population proportionally in high-income brackets. The ‘Other’ group sits around 25.1%, close to North America’s rate. This likely reflects a diverse mix of regions not explicitly listed.

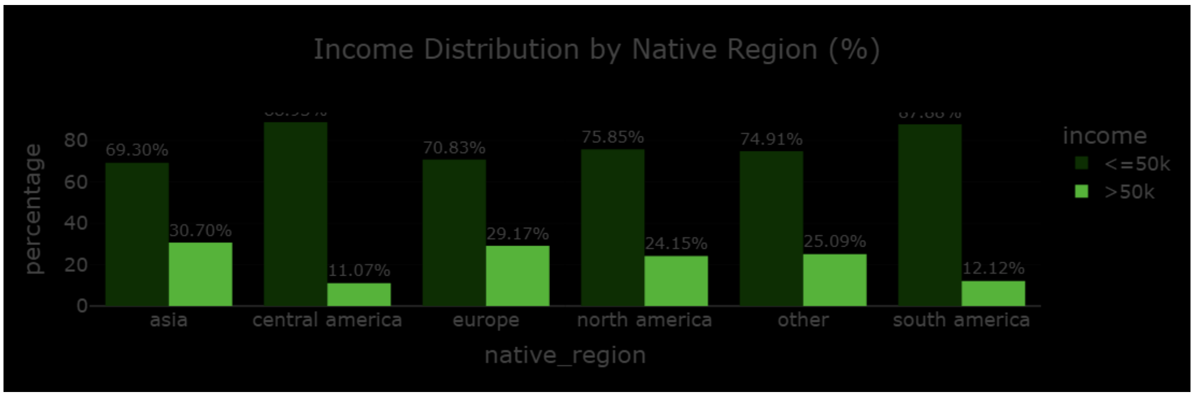
```
fig = px.bar(
    adult_df_income_age,
    x = 'age_group',
    y = 'percentage',
    color = 'income',
    title='Incoome Distribution by Age Group(%)',
    barmode='group',
    color_discrete_sequence=['#0d2e03', '#56b33a'],
    text='percentage'
)
fig.update_traces(texttemplate = '%{text:.2f}%')
fig.update_layout(template='presentation', paper_bgcolor='rgb(0, 0, 0)', plot_bgcolor='rgb(0
fig.show()
fig.write_image(os.path.join(result_dir, 'income_distribution_by_agegroup_bar_plot.jpg'))
fig.write_image(os.path.join(result_dir, 'income_distribution_by_agegroup_bar_plot.png'))
fig.write_html(os.path.join(result_dir, 'income_distribution_by_agegroup_bar_plot.html'))
```




```
WARNING Thread(Thread-75 (run)) Task(Task-1275) choreographer.browser_async:browser_async.py
```

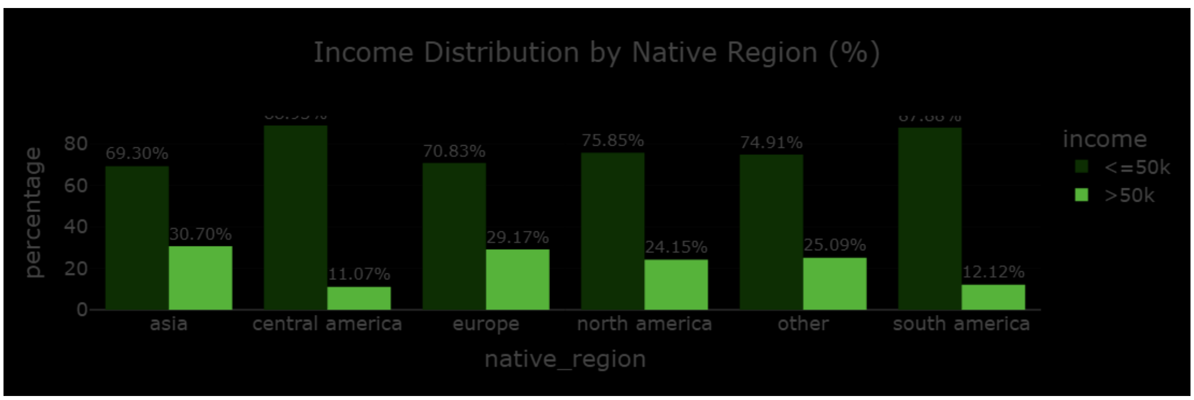
```
# Group by native_region and income, count occurrences
adult_df_income_reg = adult_df.groupby(['native_region', 'income']).size().reset_index(name=

# Calculate total per native_region
total_per_group = adult_df_income_reg.groupby('native_region')['total_income_reg'].transform

# Calculate percentage
adult_df_income_reg['percentage'] = (adult_df_income_reg['total_income_reg'] / total_per_grou

# Plot the bar chart
```

```
fig = px.bar(
    adult_df_income_reg,
    x='native_region',
    y='percentage',
    color='income',
    title='Income Distribution by Native Region (%)',
    barmode='group',
    color_discrete_sequence=['#0d2e03', '#56b33a'],
    text='percentage'
)

# Format the text on bars
fig.update_traces(texttemplate='%{text:.2f}%', textposition='outside')
fig.update_layout(template='presentation', paper_bgcolor='rgb(0, 0, 0)', plot_bgcolor='rgb(0
fig.show()
fig.write_image(os.path.join(result_dir, 'income_distribution_by_race_bar_plot.jpg'))
fig.write_image(os.path.join(result_dir, 'income_distribution_by_race_bar_plot.png'))
fig.write_html(os.path.join(result_dir, 'income_distribution_by_race_bar_plot.html'))
```




```
# Group by native_region and income, count occurrences
adult_df_income_reg = adult_df.groupby(['native_region', 'income']).size().reset_index(name=

# Calculate total per native_region
total_per_group = adult_df_income_reg.groupby('native_region')['total_income_reg'].transform

# Calculate percentage
adult_df_income_reg['percentage'] = (adult_df_income_reg['total_income_reg'] / total_per_grou

# Plot the bar chart
fig = px.bar(
```

```
    adult_df_income_reg,
    x='native_region',
    y='percentage',
    color='income',
    title='Income Distribution by Native Region (%)',
    barmode='group',
    color_discrete_sequence=['#0d2e03', '#56b33a'],
    text='percentage'
)

# Format the text on bars
fig.update_traces(texttemplate='%{text:.2f}%', textposition='outside')
fig.update_layout(template='presentation', paper_bgcolor='rgb(0, 0, 0)', plot_bgcolor='rgb(0
fig.show()
fig.write_image(os.path.join(result_dir, 'income_distribution_by_race_bar_plot.jpg'))
fig.write_image(os.path.join(result_dir, 'income_distribution_by_race_bar_plot.png'))
fig.write_html(os.path.join(result_dir, 'income_distribution_by_race_bar_plot.html'))
```




```
# Group by native_region and income, count occurrences
adult_df_income_reg = adult_df.groupby(['race', 'income']).size().reset_index(name='total_inc

# Calculate total per native_region
total_per_group = adult_df_income_reg.groupby('race')['total_income_reg'].transform('sum')

# Calculate percentage
adult_df_income_reg['percentage'] = (adult_df_income_reg['total_income_reg'] / total_per_grou

# Plot the bar chart
fig = px.bar(
    adult_df_income_reg,
```

```
    x='race',
    y='percentage',
    color='income',
    title='Income Distribution by Race (%)',
    barmode='group',
    color_discrete_sequence=['#0d2e03', '#56b33a'],
    text='percentage'
)

# Format the text on bars
fig.update_traces(texttemplate='%{text:.2f}%', textposition='outside')
fig.update_layout(template='presentation', paper_bgcolor='rgb(0, 0, 0)', plot_bgcolor='rgb(0
fig.show()
fig.write_image(os.path.join(result_dir, 'income_distribution_by_race_bar_plot.jpg'))
fig.write_image(os.path.join(result_dir, 'income_distribution_by_race_bar_plot.png'))
fig.write_html(os.path.join(result_dir, 'income_distribution_by_race_bar_plot.html'))
```


```
WARNING Thread(Thread-83 (run)) Task(Task-1435) choreographer.browser_async:browser_async.py
```

```
# Group by native_region and income, count occurrences
adult_df_income_reg = adult_df.groupby(['race', 'income']).size().reset_index(name='total_inc

# Calculate total per native_region
total_per_group = adult_df_income_reg.groupby('race')['total_income_reg'].transform('sum')

# Calculate percentage
adult_df_income_reg['percentage'] = (adult_df_income_reg['total_income_reg'] / total_per_grou

# Plot the bar chart
```

```python
fig = px.bar(
    adult_df_income_reg,
    x='race',
    y='percentage',
    color='income',
    title='Income Distribution by Race (%)',
    barmode='group',
    color_discrete_sequence=['#0d2e03', '#56b33a'],
    text='percentage'
)

# Format the text on bars
fig.update_traces(texttemplate='%{text:.2f}%', textposition='outside')
fig.update_layout(template='presentation', paper_bgcolor='rgb(0, 0, 0)', plot_bgcolor='rgb(0
fig.show()
fig.write_image(os.path.join(result_dir, 'income_distribution_by_race_bar_plot.jpg'))
fig.write_image(os.path.join(result_dir, 'income_distribution_by_race_bar_plot.png'))
fig.write_html(os.path.join(result_dir, 'income_distribution_by_race_bar_plot.html'))
```


```
WARNING Thread(Thread-83 (run)) Task(Task-1435) choreographer.browser_async:browser_async.py
```

```python
adult_df_income_edu_occ = adult_df.groupby(['education_level', 'income','occupation-grouped']
adult_df_income_edu_occ
```

| | education_level | income | occupation-grouped | total |
|---|---|---|---|---|
| 60 | secondary graduate | <=50k | white collor | 2617 |
| 95 | unemployed | <=50k | white collor | 2497 |
| 54 | secondary graduate | <=50k | blue collar | 1871 |

| | education_level | income | occupation-grouped | total |
|---|---|---|---|---|
| 86 | tertiary | >50k | white collor | 1865 |
| 77 | tertiary | <=50k | white collor | 1667 |
| ... | ... | ... | ... | ... |
| 39 | secondary | <=50k | military | 1 |
| 33 | primary | >50k | service | 1 |
| 21 | preschool | <=50k | white color | 1 |
| 82 | tertiary | >50k | military | 1 |
| 73 | tertiary | <=50k | military | 1 |

```
final_file = os.path.join(processed_dir, 'adult_cleaned.csv')
adult_df.to_csv(final_file, index=False)
```