Certainly! I’ll break down each part of the code and explain what each section does. This setup is to perform an Exploratory Data Analysis (EDA) on the earthquake dataset, following the steps your instructor outlined.

---

### 1. Set Up the Notebook and Load Data

In Jupyter Notebook, using special markers like `# %% [markdown]` and `# %%` helps to organize the notebook into sections. The `# %% [markdown]` line indicates a markdown cell, which is where you can add explanations, titles, or descriptions without running code. The `# %%` line indicates a code cell.

```python

# %% [markdown]

# # Exercise 1: Earthquake Data Analysis

# ## Manual Exploration of Data

```

This is a \*\*markdown cell\*\* where you write text to describe what the notebook or section is doing. Here, it's giving a title ("Exercise 1: Earthquake Data Analysis") and a subheading ("Manual Exploration of Data").

---

### 2. Import Libraries

```python

# %% Import libraries

import pandas as pd

import matplotlib.pyplot as plt

from ydata\_profiling import ProfileReport

```

This cell imports the libraries needed for the analysis:

- \*\*pandas\*\*: A powerful library for data manipulation and analysis, especially for working with structured data in tables (DataFrames).

- \*\*matplotlib.pyplot\*\*: A plotting library used for creating static, interactive, and animated visualizations. Here, we’ll use it to create plots to understand data distributions.

- \*\*ydata\_profiling\*\*: A library for generating automated EDA reports. The `ProfileReport` function from `ydata\_profiling` can create an HTML report that summarizes the dataset with just a few lines of code.

---

### 3. Load the Data

```python

# %% Load data

FILEPATH = '../Data/raw/earthquake\_data.csv'

df = pd.read\_csv(FILEPATH)

df.head()

```

- \*\*FILEPATH\*\*: This variable stores the path to the dataset file. Here, `../Data/raw/earthquake\_data.csv` points to a file in the `Data/raw` directory.

- \*\*pd.read\_csv(FILEPATH)\*\*: This reads a CSV file into a DataFrame. `pd.read\_csv` is a function in pandas specifically for reading CSV files.

- \*\*df.head()\*\*: This command displays the first five rows of the DataFrame `df`. It's useful for quickly checking the structure and content of the dataset right after loading.

---

### 4. Inspect and Drop Columns

```python

# %% Check column names

df.columns

# %% Display information about data

df.info()

# %% Drop specified columns

df = df.drop(columns=['title', 'continent', 'alert', 'location', 'country'])

df.head()

```

These cells are for data inspection and cleanup:

- \*\*df.columns\*\*: This command displays the names of all columns in the DataFrame. It’s helpful to see which columns are present.

- \*\*df.info()\*\*: This provides a summary of the DataFrame, showing the number of entries, data types for each column, and counts of non-null values.

- \*\*df.drop(columns=...)\*\*: This line removes specified columns from the DataFrame. In your exercise, you were asked to remove columns like `'title'`, `'continent'`, `'alert'`, `'location'`, and `'country'` as they are likely irrelevant for the analysis. `df.head()` displays the first few rows of the modified DataFrame.

---

### 5. Check for Missing Values and Drop Rows with Missing Data

```python

# %% Check for missing values

df.isnull().sum()

# %% Drop rows with missing values

df = df.dropna()

df.head()

# %% Check new shape

df.shape

```

- \*\*df.isnull().sum()\*\*: This command counts missing values in each column. It’s essential to know if there are missing values so you can decide whether to fill them in or drop them.

- \*\*df.dropna()\*\*: This removes rows that contain any missing values. After dropping these rows, you may use `df.head()` to inspect the data.

- \*\*df.shape\*\*: This returns the dimensions of the DataFrame as `(rows, columns)`. It’s useful for seeing the impact of dropping rows with missing values on the dataset size.

---

### 6. Check Label Balance (If Applicable)

```python

# %% Check balance of target labels if available (replace 'label\_column' with the actual column name)

df["label\_column"].value\_counts() # Adjust 'label\_column' as necessary

```

This cell would be used to check the balance of classes in a target label column. \*\*Note\*\*: In your current dataset, you may not have a specific label column yet, so you can skip or adapt this step if you're not focusing on classification.

- \*\*df["label\_column"].value\_counts()\*\*: This command counts the occurrences of each unique value in a specified column (`label\_column`). This helps check if the dataset is balanced or imbalanced with respect to the target labels.

---

### 7. Descriptive Statistics for Numerical Features

```python

# %% Descriptive statistics for numeric features

pd.set\_option('display.float\_format', '{:.2f}'.format)

numeric\_features = df.select\_dtypes(include=["number"]).columns

df[numeric\_features].describe()

```

- \*\*pd.set\_option\*\*: This changes the display format for floating-point numbers to 2 decimal places, which can make tables easier to read.

- \*\*df.select\_dtypes(include=["number"]).columns\*\*: This selects columns with numerical data types.

- \*\*df[numeric\_features].describe()\*\*: This command provides summary statistics (like mean, median, min, max) for all numerical columns.

---

### 8. Automated EDA with ydata-profiling

```python

# %% [markdown]

# ## Automated EDA using ydata-profiling

# %% Prepare data for ydata-profiling

dfy = df.copy()

dfy["label\_column"] = dfy["label\_column"].astype(str) # Change 'label\_column' to your label's name if needed

# %% Generate the profiling report

profile = ProfileReport(dfy, title="Profiling Report - Earthquake Data")

profile.to\_file("earthquake\_data\_profiling.html")

```

This section uses \*\*ydata-profiling\*\* to generate a quick automated report on the dataset.

- \*\*df.copy()\*\*: Creates a copy of `df` as `dfy`, which can be used for generating the report without altering the original data.

- \*\*dfy["label\_column"].astype(str)\*\*: This converts the target label column to a string type, making sure it’s interpreted as categorical data.

- \*\*ProfileReport(dfy, title=...)\*\*: Creates a profiling report on `dfy` with a given title.

- \*\*profile.to\_file("earthquake\_data\_profiling.html")\*\*: Saves the generated report as an HTML file, which you can open in a browser for easy viewing.

---

### 9. Outlier Detection and Handling

```python

# %% [markdown]

# ## Outliers Detection

# %% Visualize distributions

df[numeric\_features].hist(figsize=(12, 10))

plt.show()

```

- \*\*df[numeric\_features].hist\*\*: Creates histograms for each numeric feature in the DataFrame. Histograms help visualize distributions and spot potential outliers.

- \*\*plt.show()\*\*: Displays the plots.

For more advanced outlier detection, you could use specific outlier detection models or rules (like values outside of 1.5 times the interquartile range), but here, we’re keeping it simple with histograms.

---

### 10. Train-Test Split

```python

# %% [markdown]

# ## Train-Test Split

from sklearn.model\_selection import train\_test\_split

FEATURES = ['magnitude', 'date\_time', 'cdi', 'mmi', 'sig', 'net', 'nst', 'dmin', 'gap', 'magType', 'depth', 'latitude', 'longitude']

LABEL = 'tsunami' # Replace 'tsunami' with the actual label column if applicable

# Define X and y

X = df[FEATURES]

y = df[LABEL]

# Perform train-test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=42, stratify=y)

# Display split results

X\_train.shape, X\_test.shape, y\_train.value\_counts(normalize=True), y\_test.value\_counts(normalize=True)

```

This section splits the data into training and testing sets.

- \*\*train\_test\_split\*\*: Splits `X` (features) and `y` (target) into training and testing sets. Here, `test\_size=0.25` means 25% of the data is reserved for testing.

- \*\*stratify=y\*\*: Ensures that the split maintains the class distribution of `y`, which is useful for classification tasks.

---

This setup covers the core steps of EDA and prepares data for further analysis or modeling. Let me know if you have more questions about each step!