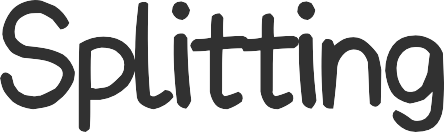
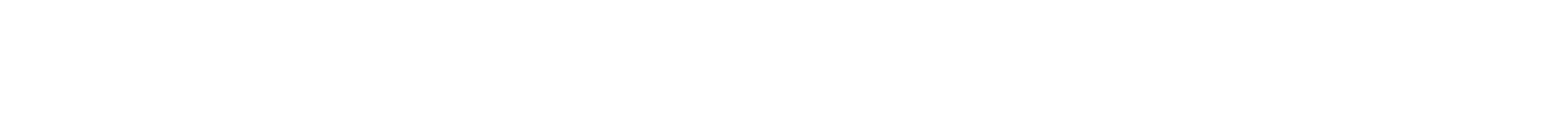
Earthquake Prediction Model Using Python.

Visualizing the data on a world map



And

Splitting it into training and testing Sets.

AI-Phase 4 Project

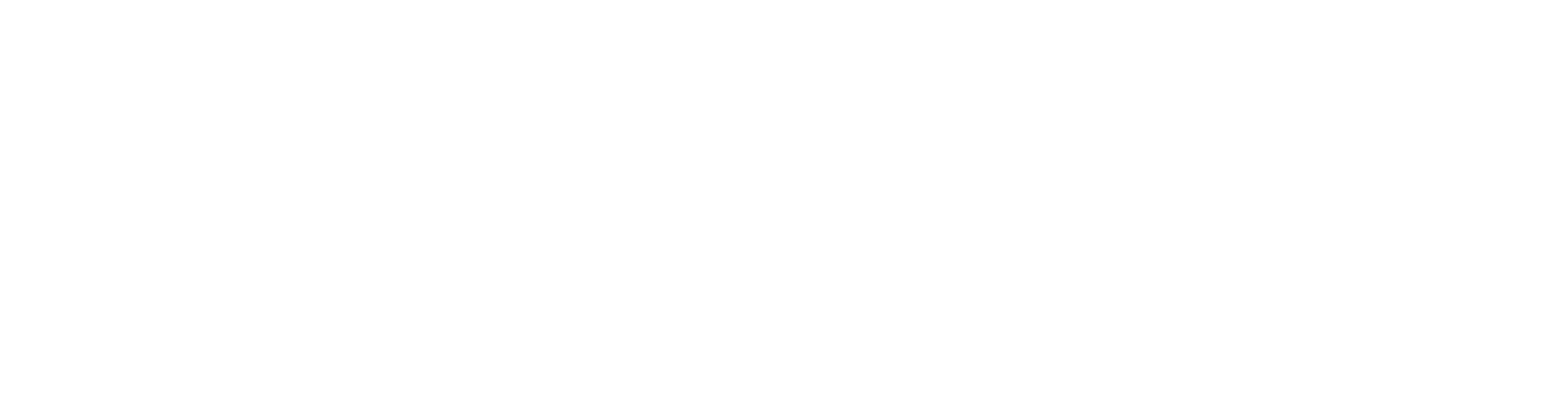
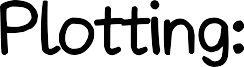
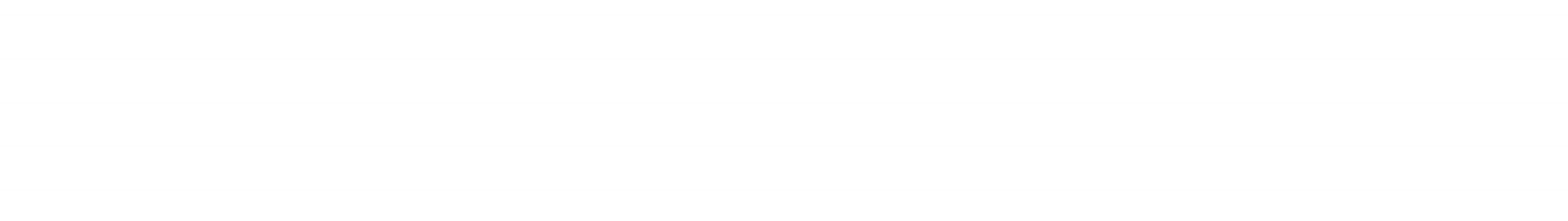
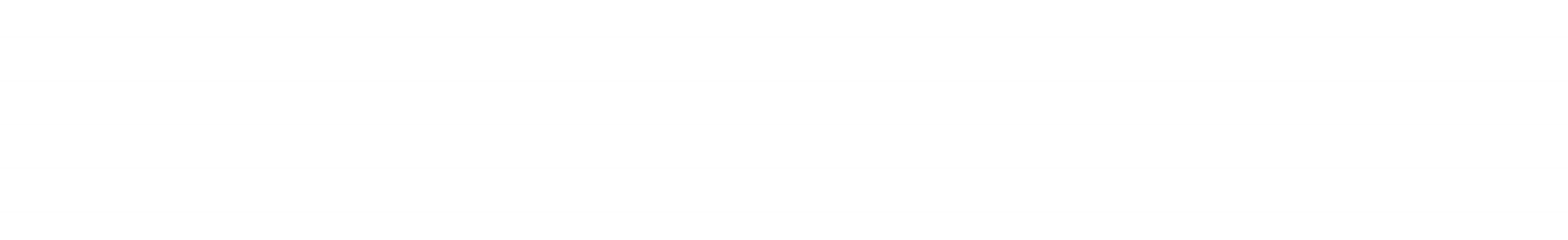
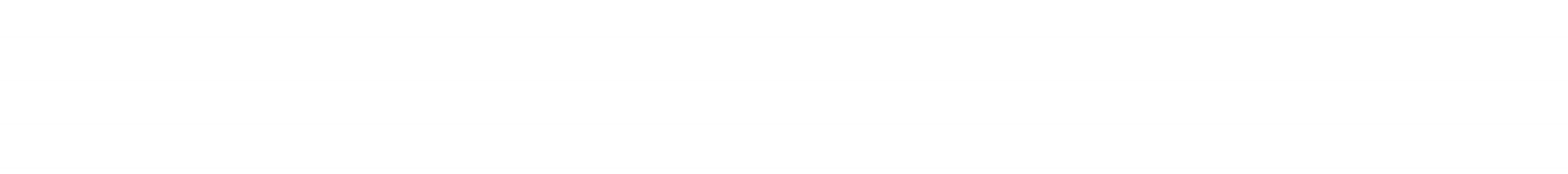
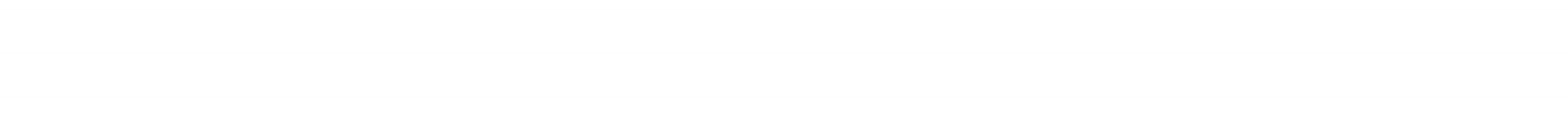
Selvam College Of Technology

Visualization:



Visualizing data on a world map and splitting it into training and testing sets are essential steps in data analysis and machine learning. Let's break down these tasks:

Libraries:



# Visualizing Data on a World Map:

You can use Python along with libraries like Pandas, Matplotlib, and Plotly for data visualization. For world maps specifically, the geopandas library is very useful as it extends Pandas to allow spatial operations.

Steps:

Data Collection: First, you need data with geographical information. For example, if you have a dataset with countries and some associated data, you can use it.

Data Preparation: Load your data into a Pandas DataFrame. Ensure it contains columns like country names, numerical values for visualization, and possibly latitude and longitude for precise location-based plotting.

Map Plotting:

Use geopandas to load a world map shapefile.

Merge your data with the shapefile based on common keys (like country names).

Plot the map using Matplotlib or Plotly, coloring countries based on the data values you want to visualize.

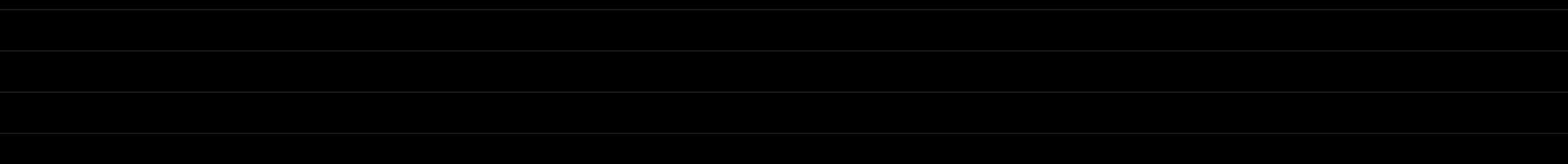
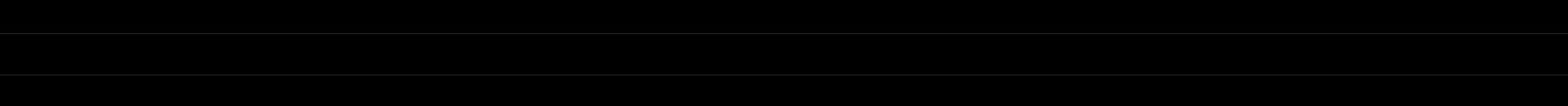
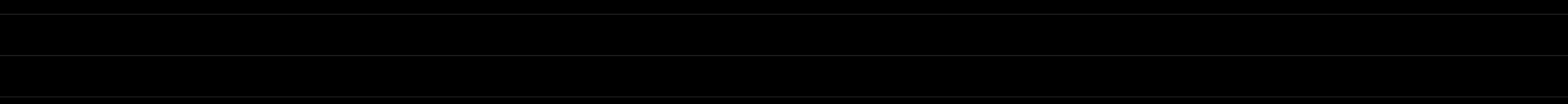
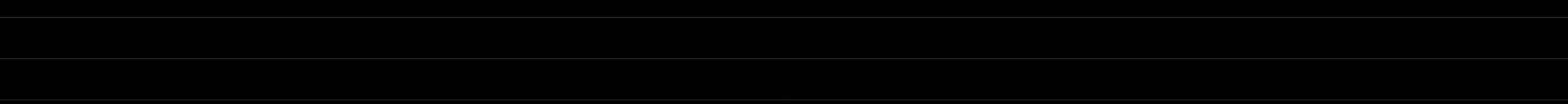
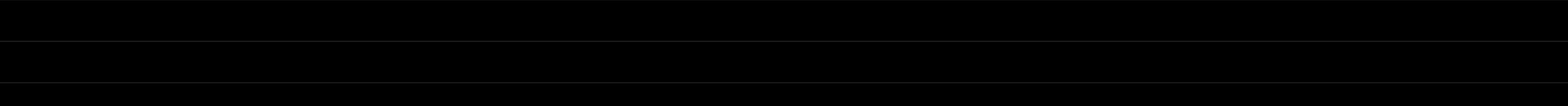
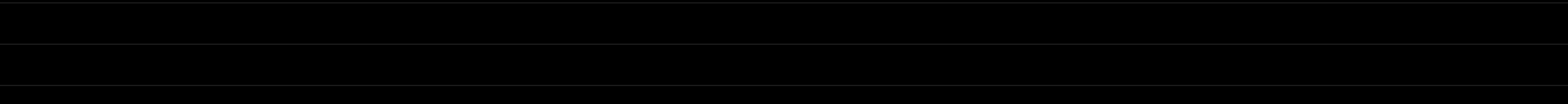
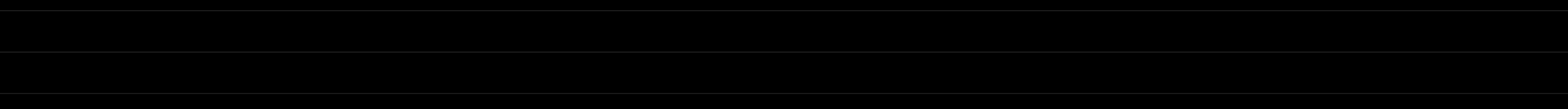
The process of transcribing weather information onto maps, diagrams, etc.



It usually refers specifically to decoding synoptic reports and entering those data in conventional station-model form on synoptic charts. It is done either

manually or by computer.

Example code snippet using Geopandas and Matplotlib:



import geopandas as gpd import matplotlib.pyplot as plt # Load world map shapefile

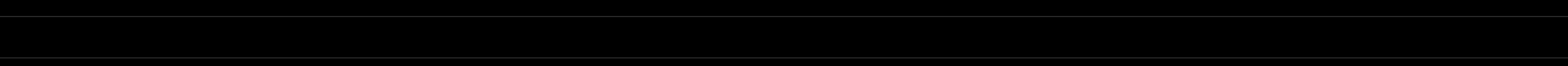
world = gpd.read\_file(gpd.datasets.get\_path('naturalearth\_lowres')) # Merge your data with the world map data

merged = world.set\_index('name').join(your\_data.set\_index('CountryName')) # Plotting

fig, ax = plt.subplots(1, 1, figsize=(15, 10))

merged.plot(column='YourDataColumn', cmap='coolwarm', linewidth=0.8, ax=ax, edgecolor='0.8', legend=True)

plt.show()



# Splitting Data into Training and Testing Sets:

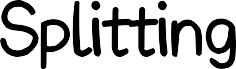


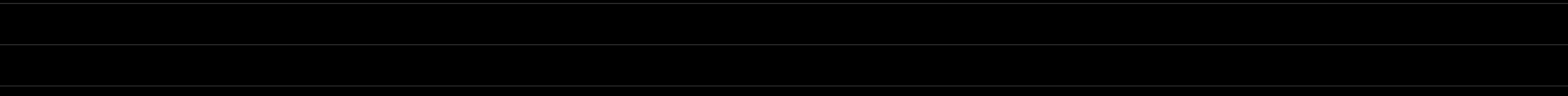
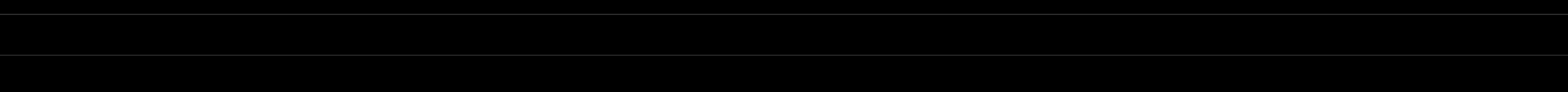
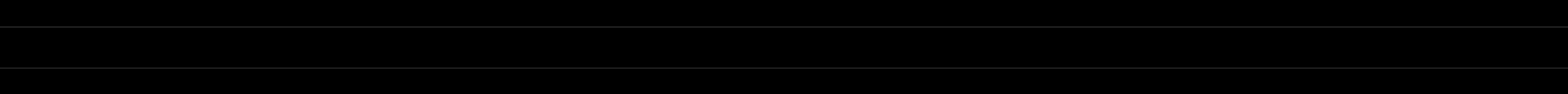
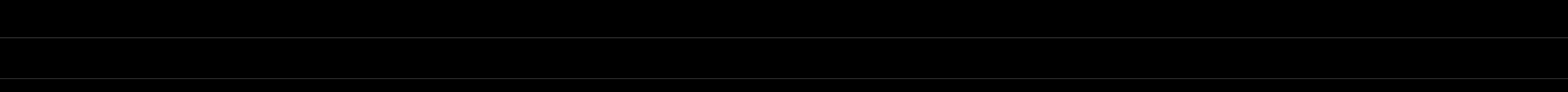
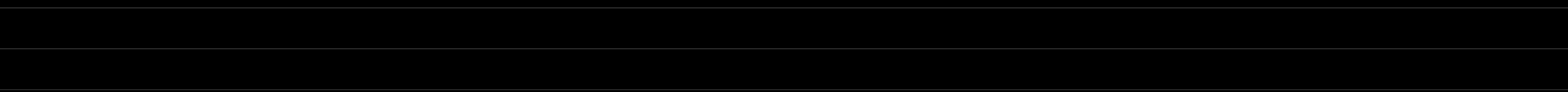
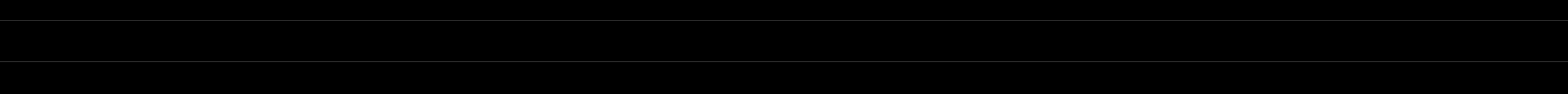
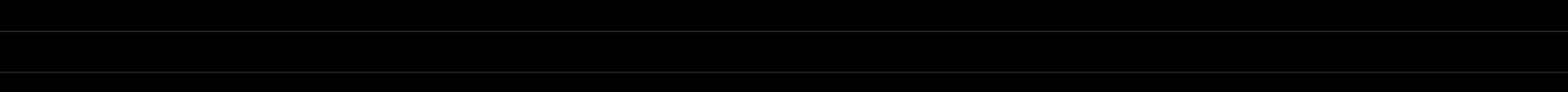
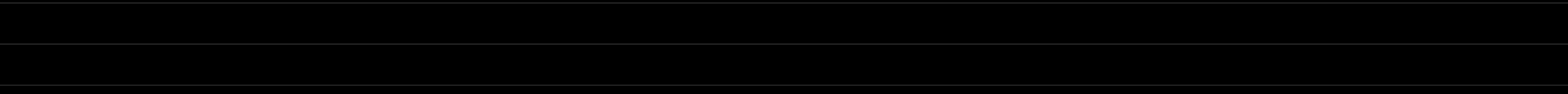
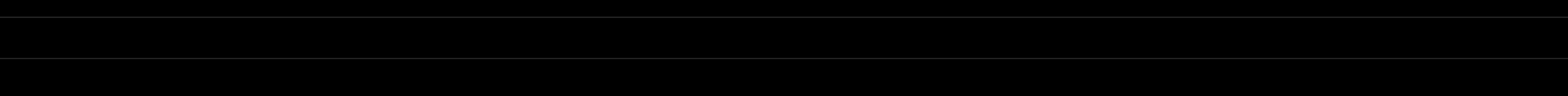
Libraries:

For splitting data into training and testing sets, you can use the train\_test\_split function from the sklearn.model\_selection module.

Steps:

Data Preparation: Ensure your data is cleaned and formatted properly. Convert categorical variables to numerical if needed.

Splitting Data:



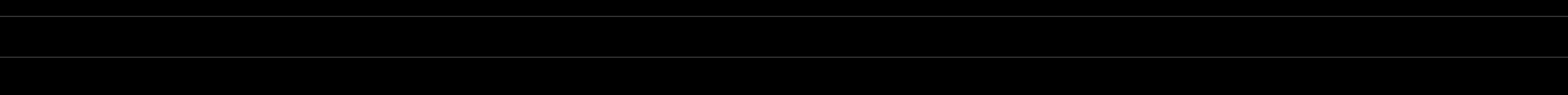
Separate your features (X) from the target variable (y).

Use train\_test\_split to split the data into training and testing sets. Example code snippet for splitting data:

from sklearn.model\_selection import train\_test\_split # X contains features, y contains labels

X = your\_data.drop(columns=['target\_column']) y = your\_data['target\_column']

# Split data into training and testing sets (80% train, 20% test)



X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

Now you have X\_train, X\_test, y\_train, and y\_test which you can use for training and evaluating your machine learning models.

Make sure to replace 'YourDataColumn' with the actual column name you want to visualize, and 'target\_column' with the column name representing your target variable.

# Tools for Visualizing Data on a World Map:



## Google Maps API:



Google Maps provides a robust API for integrating maps into web applications. You can customize the maps and add data layers on top of them.

## Leaflet:

Leaflet is an open-source JavaScript library for interactive maps. It's lightweight, mobile-friendly, and provides a great platform for creating customized maps.

## Mapbox:

Mapbox offers a powerful platform for customizing maps and integrating them into web and mobile applications. It provides APIs and libraries for creating visually appealing maps.

## Tableau:

Tableau is a popular data visualization tool that supports geographical visualizations. It allows you to create interactive dashboards with maps and various data visualization components.

## D3.js:



D3.js is a JavaScript library for creating dynamic, interactive data visualizations in web browsers. It's highly flexible and can be used to create custom map visualizations.

# Tools for Splitting Data into Training and Testing Sets:



## scikit-learn:

Scikit-learn is a powerful Python library for machine learning. It provides the train\_test\_split function for easily splitting data into training and testing sets. Scikit-learn also offers various machine learning algorithms for model training and evaluation.

## Pandas:

Pandas is a Python library for data manipulation and analysis. It provides powerful data structures like DataFrames, which can be used to split data into training and testing sets.

## Excel/Google Sheets:

For smaller datasets, you can manually split data into training and testing sets using spreadsheet software like Excel or Google Sheets. While this method is not scalable for large datasets, it can be useful for educational purposes or small projects.

## NumPy:



NumPy is a Python library for numerical computing. It can be used to perform random sampling and splitting of arrays, which is useful when dealing with numerical data in machine learning.

## RapidMiner:

RapidMiner is a data science platform that provides a visual environment for building machine learning models. It offers various data preprocessing tools, including splitting data into training and testing sets.

When choosing a tool, consider factors such as the size of your dataset, your programming language preference, and the level of customization and interactivity you require in your visualizations.

## Visualizing Data on a World Map using Leaflet(Python):

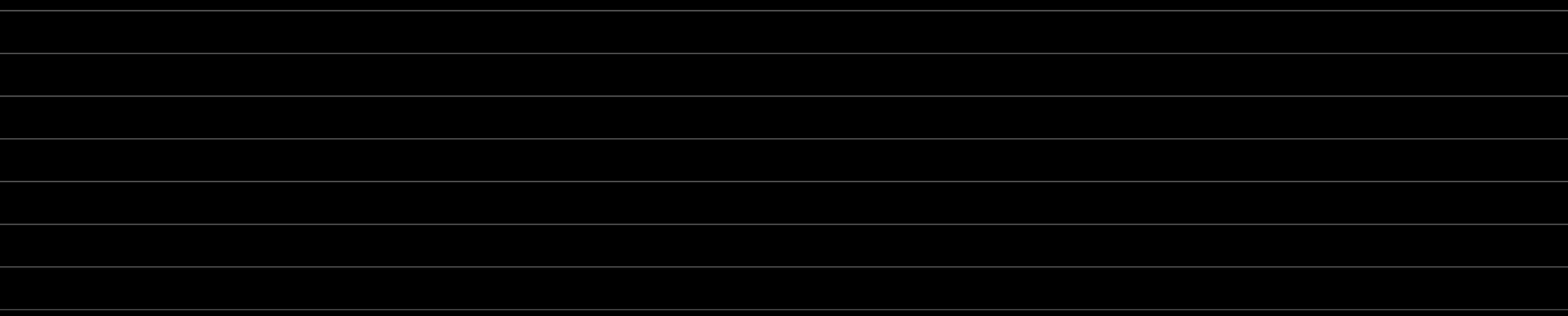


To visualize data on a world map using Leaflet in a Python environment, you can utilize the folium library, which allows you to create Leaflet maps easily.

Here's an example of how you can visualize data on a world map using folium in Python:

## Prerequisites:

Make sure you have the folium library installed. You can install it using pip if you haven't already:



pip install folium

Python Code to Visualize Data on a World Map using Leaflet:

Python Code:



import folium

# Sample data with latitude, longitude, and a value data = [

{"name": "New York", "location": [40.7128, -74.0060], "value": 100},

{"name": "London", "location": [51.5074, -0.1278], "value": 200}, # Add more data points as needed

]



# Create a map centered at [0, 0] with zoom level 2

world\_map = folium.Map(location=[0, 0], zoom\_start=2)

# Add markers to the map based on the data for city in data:

folium.Marker(location=city["location"], popup=f'Value:

{city["value"]}').add\_to(world\_map)

# Save the map as an HTML file world\_map.save("world\_map.html")

# Optionally, you can also display the map in a Jupyter Notebook # world\_map

In this example, the folium library is used to create a map (world\_map). The data list contains dictionaries with city names, latitude and longitude coordinates, and corresponding values.

The folium.Marker function is used to add markers for each city on the map.

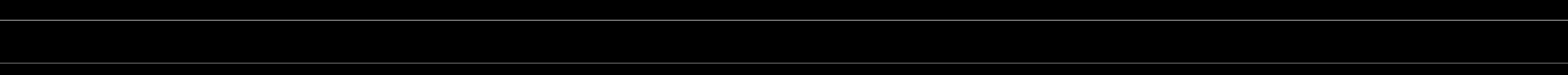
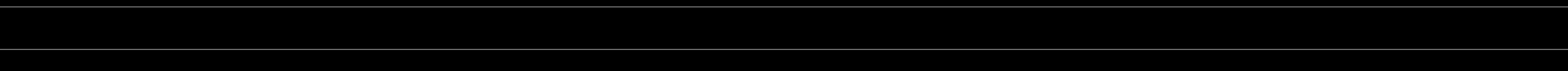
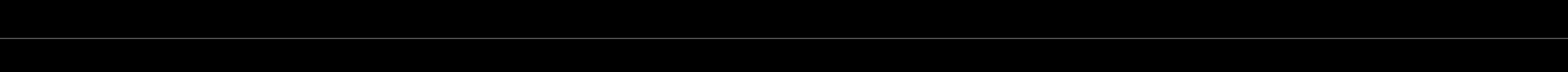
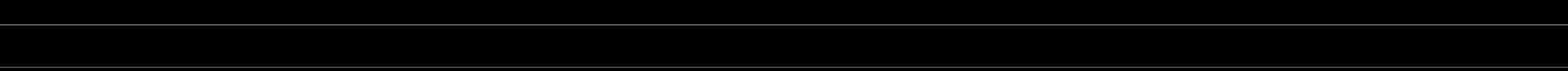
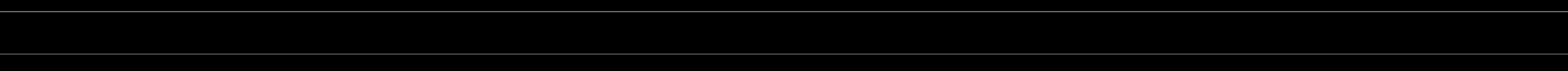
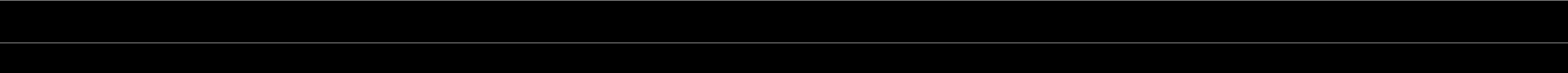
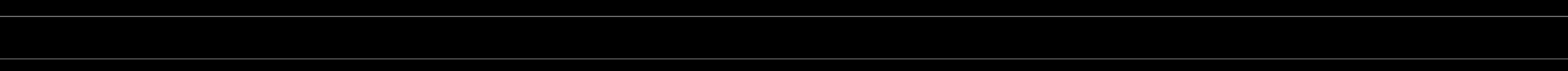
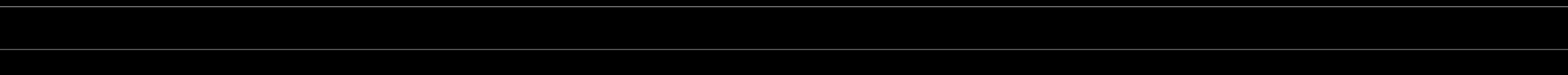
The resulting map is saved as an HTML file (world\_map.html), but you can also display it directly in a Jupyter Notebook by uncommenting the world\_map line.

Remember to customize the data list with your specific data points and their geographical coordinates. This script will generate an interactive map where you can click on markers to view the associated values.

## Splitting Data into Training and Testing Sets using scikit-learn (Python):



Python Code:



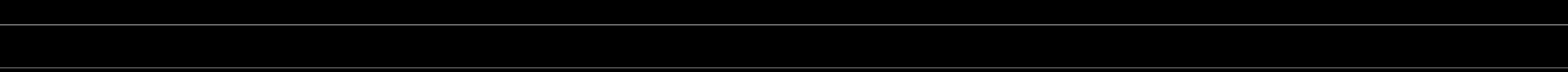
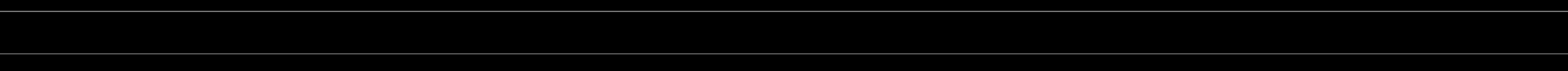
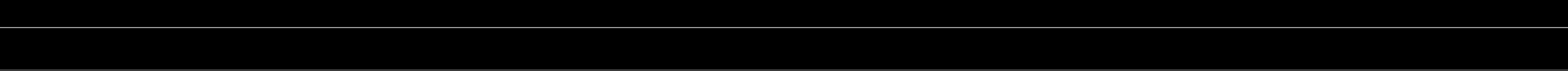
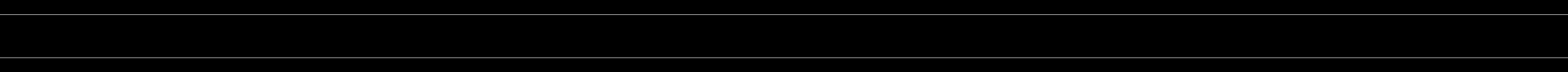
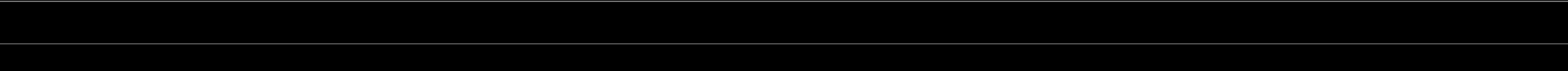
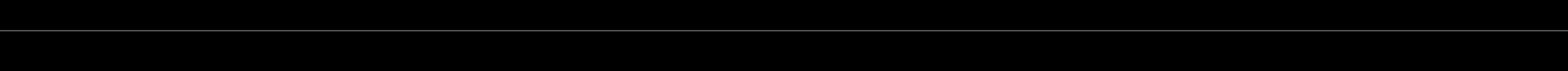
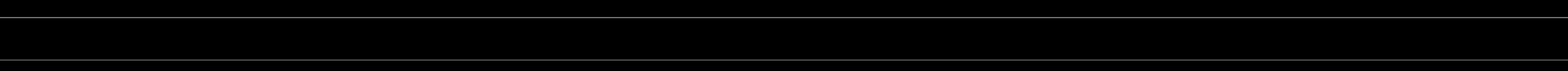
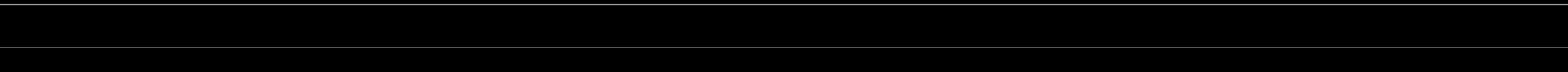
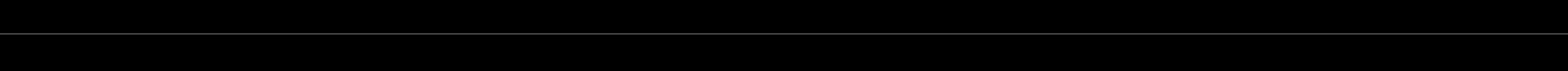
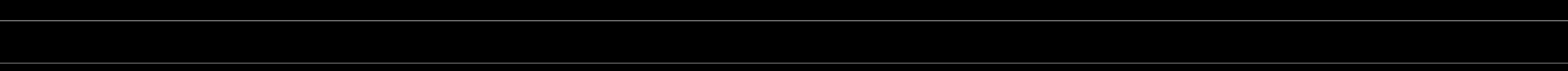
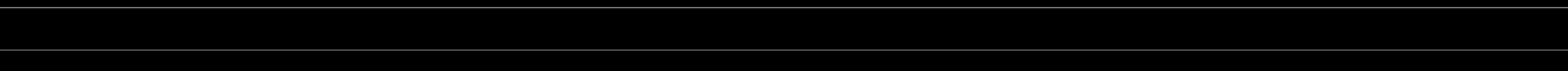
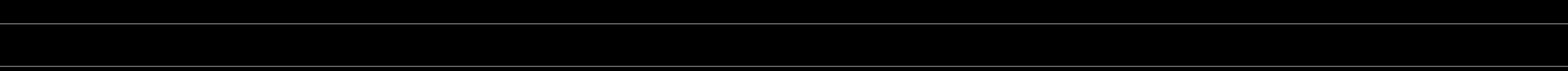
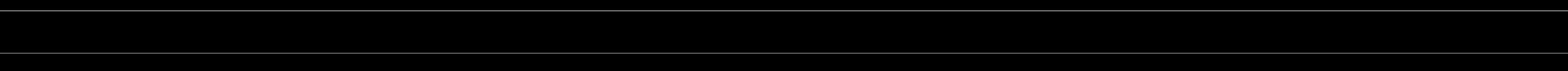
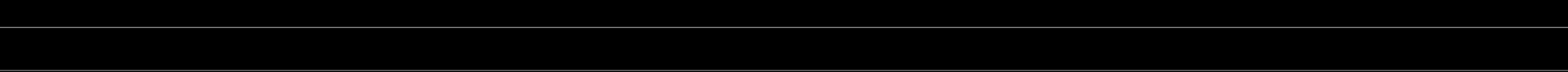
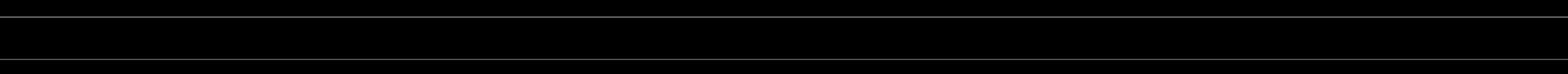
# Import necessary libraries

from sklearn.model\_selection import train\_test\_split import pandas as pd

# Sample data: features (X) and target variable (y) data = {

'Feature1': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],

'Feature2': [11, 12, 13, 14, 15, 16, 17, 18, 19, 20],



'Target': [0, 1, 0, 1, 1, 0, 1, 0, 1, 0]

}

# Create a DataFrame df = pd.DataFrame(data)

# Split data into features (X) and target variable (y) X = df[['Feature1', 'Feature2']]

y = df['Target']

# Split data into training and testing sets (80% train, 20% test)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Print the shapes of the resulting sets print("X\_train shape:", X\_train.shape) print("X\_test shape:", X\_test.shape) print("y\_train shape:", y\_train.shape) print("y\_test shape:", y\_test.shape)

In this Python example, scikit-learn's train\_test\_split function is used to split the dataset into training and testing sets. The data consists of two features ('Feature1' and 'Feature2') and a binary target variable ('Target').

The test\_size parameter determines the proportion of the dataset that will be included in the test split (in this case, 20%).

These examples demonstrate how to visualize data on a world map using Leaflet (JavaScript) and how to split data into training and testing sets using scikit-learn (Python).