Guide Document to Use ‘ipywidgets’ Dropdown Filter in Folium Mapping

This document is designed to provide a comprehensive guide on using the ‘ipywidgets’ dropdown filter in Jupyter Notebooks to create interactive maps with Folium. By the end of this guide, you will be able to filter data dynamically on a map using a dropdown menu. This is particularly useful for visualizing geographic data and providing an interactive experience.

1. What You Will Learn

- Setting Up Your Environment
- Loading and Preparing Data
- Creating a Basic Map with Folium
- Adding a Dropdown Filter with ‘ipywidgets’
- Linking the Dropdown to the Map
- Customizing the Map and Dropdown
- Deploying and Sharing Your Map

2. Setting Up Your Environment

Before we begin, ensure that you have the required packages installed in your environment. You will need:

- Folium for mapping
- ‘ipywidgets’ for the dropdown filter

You can install these packages using the following commands:

```
[1]: !pip install folium
     !pip install ipywidgets
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: folium in c:\users\venuk\appdata\roaming\python\python311\site-packages (0.16.0)
Requirement already satisfied: branca>=0.6.0 in c:\users\venuk\appdata\roaming\python\python311\site-packages (from folium) (0.7.1)
Requirement already satisfied: jinja2>=2.9 in c:\programdata\anaconda3\lib\site-packages (from folium) (3.1.3)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-packages (from folium) (1.26.4)
Requirement already satisfied: requests in c:\programdata\anaconda3\lib\site-packages (from folium) (2.31.0)
Requirement already satisfied: xyzservices in c:\programdata\anaconda3\lib\site-packages (from folium) (2022.9.0)
Requirement already satisfied: MarkupSafe>=2.0 in c:\programdata\anaconda3\lib\site-packages (from jinja2>=2.9->folium) (2.1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\programdata\anaconda3\lib\site-packages (from requests->folium) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\programdata\anaconda3\lib\site-packages (from requests->folium) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\programdata\anaconda3\lib\site-packages (from requests->folium) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in c:\programdata\anaconda3\lib\site-packages (from requests->folium) (2024.2.2)
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: ipywidgets in c:\users\venuk\appdata\roaming\python\python311\site-packages (8.1.5)
Requirement already satisfied: comm>=0.1.3 in c:\users\venuk\appdata\roaming\python\python311\site-packages (from ipywidgets) (0.2.2)
Requirement already satisfied: ipython>=6.1.0 in c:\programdata\anaconda3\lib\site-packages (from ipywidgets) (8.20.0)
Requirement already satisfied: traitlets>=4.3.1 in c:\programdata\anaconda3\lib\site-packages (from ipywidgets) (5.7.1)
Requirement already satisfied: widgetsnbextension~=4.0.12 in c:\users\venuk\appdata\roaming\python\python311\site-packages (from ipywidgets) (4.0.13)
```

```
[ ]:
```

3. Loading and Preparing Data

Let's start by loading the data you'll use for mapping. For this guide, we'll use a dataset containing locations of cafes and restaurants.

Example Dataset Structure

Assume you have a CSV file with the following columns:

- trading_name: Name of the business (e.g., "Cafe Good")
- latitude: Latitude coordinate (e.g., -37.8136)
- longitude: Longitude coordinate (e.g., 144.9631)
- seating_capacity: Number of seats available at the location

Here's how you can load the data:

[8]:
```
import pandas as pd

# Load the dataset
df = pd.read_csv('cafes_and_restaurants.csv')

# Display the first few rows to verify the data
df.head(3)
```

[8]:

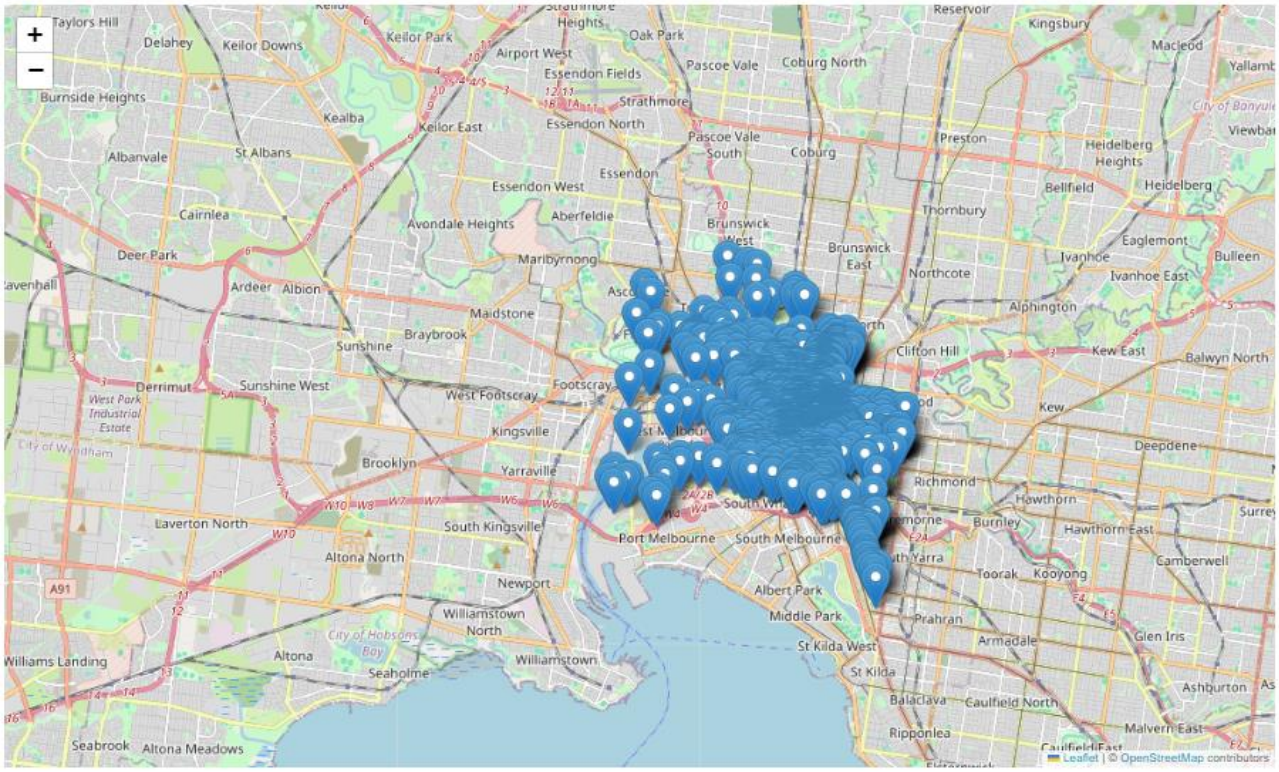
	census_year	property_id	building_address	trading_name	industry_anzsic4_description	seating_type	number_of_seats	location	latitude	longitude
0	2002	100041	253 Abbotsford Street NORTH MELBOURNE 3051	Abbot's Seafood	Takeaway Food Services	Seats - Indoor	8	-37.80132776477555, 144.9454531679633	-37.801328	144.945453
1	2002	100084	515-517 Abbotsford Street NORTH MELBOURNE 3051	La Dish Fine Foods	Takeaway Food Services	Seats - Indoor	14	-37.79420437857174, 144.94676212668315	-37.794204	144.946762
2	2011	100112	340-348 Abbotsford Street NORTH MELBOURNE 3051	Papa Guiseppe Pizza & Pasta	Takeaway Food Services	Seats - Indoor	16	-37.798727812, 144.94650694307907	-37.798728	144.946507

4. Creating a Basic Map with Folium

Next, create a basic map using Folium. Folium is a powerful Python library that makes it easy to visualize data that's been manipulated in Python on an interactive Leaflet map.

[10]:
```
import folium

# Create a base map centered around Melbourne
m = folium.Map(location=[-37.8136, 144.9631], zoom_start=12)

# Add markers for each location
for _, row in df.iterrows():
    folium.Marker(
        location=[row['latitude'], row['longitude']],
        popup=row['trading_name']
    ).add_to(m)

# Display the map
m
```

[10]:



5. Adding a Dropdown Filter with ‘ipywidgets’

To make the map interactive, we'll add a dropdown menu using ipywidgets that allows users to filter the displayed locations by business name.

Now, import ‘ipywidgets’ and create a dropdown widget:

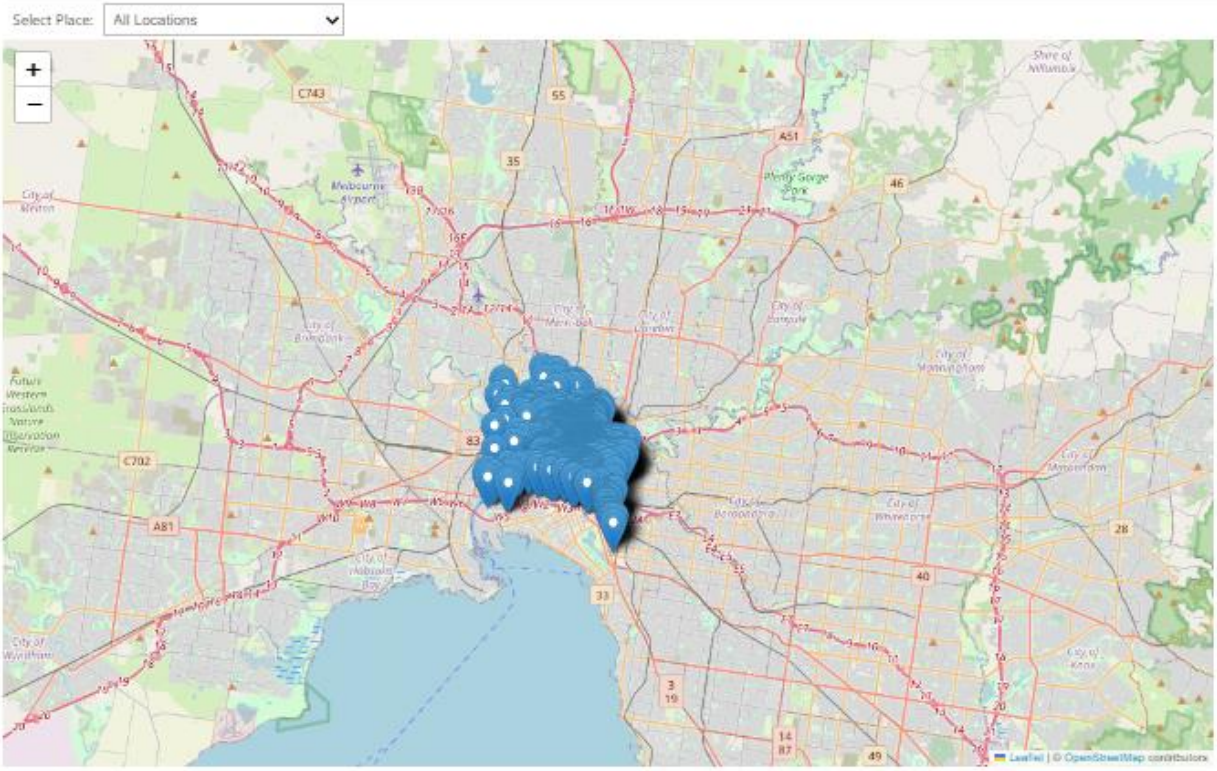
[11]:
```
import ipywidgets as widgets
from IPython.display import display

# Create a dropdown widget
dropdown = widgets.Dropdown(
    options=['All Locations'] + list(df['trading_name'].unique()),
    description='Select Place:',
)

# Display the dropdown
display(dropdown)
```

Select Place: All Locations

6. Linking the Dropdown to the Map

Now that we have a dropdown, let's link it to the map so that the map updates when a different option is selected.

[13]:
```
from IPython.display import clear_output

def update_map(trading_name):
    clear_output(wait=True)

    # Create a fresh map
    m = folium.Map(location=[-37.8136, 144.9631], zoom_start=12)

    if trading_name == "All Locations":
        for _, row in df.iterrows():
            folium.Marker(
                location=[row['latitude'], row['longitude']],
                popup=row['trading_name']
            ).add_to(m)
    else:
        selected_row = df[df['trading_name'] == trading_name].iloc[0]
        folium.Marker(
            location=[selected_row['latitude'], selected_row['longitude']],
            popup=selected_row['trading_name']
        ).add_to(m)

    # Display the updated map
    display(m)

# Connect the dropdown to the update_map function
widgets.interact(update_map, trading_name=dropdown)
```



[13]:
```
<function __main__.update_map(trading_name)>
```

Explanation:

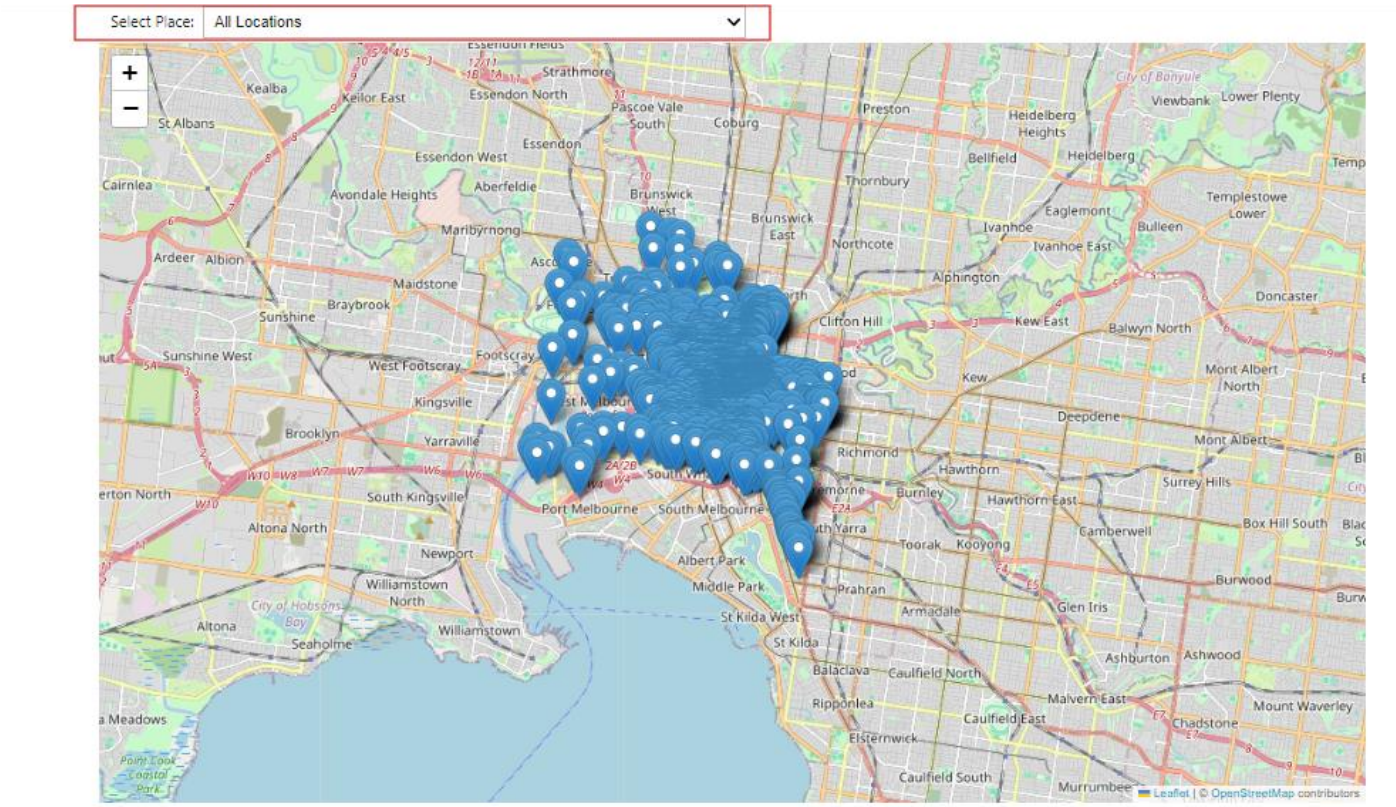
- clear_output(wait=True): Clears the previous map when a new selection is made.
- Dropdown Options: If "All Locations" is selected, all markers are displayed. Otherwise, only the selected location is shown.
- Interactive Map: The map updates in real-time based on the user's dropdown selection.

7. Customizing the Map and Dropdown

You can further customize the map and dropdown to enhance the user experience:

Customizing the Dropdown

You can adjust the appearance of the dropdown by setting the layout:




```
[6]: <function __main__.update_map(trading_name)>
```

```
[8]: dropdown.layout.width = '50%'  # Make the dropdown wider
```

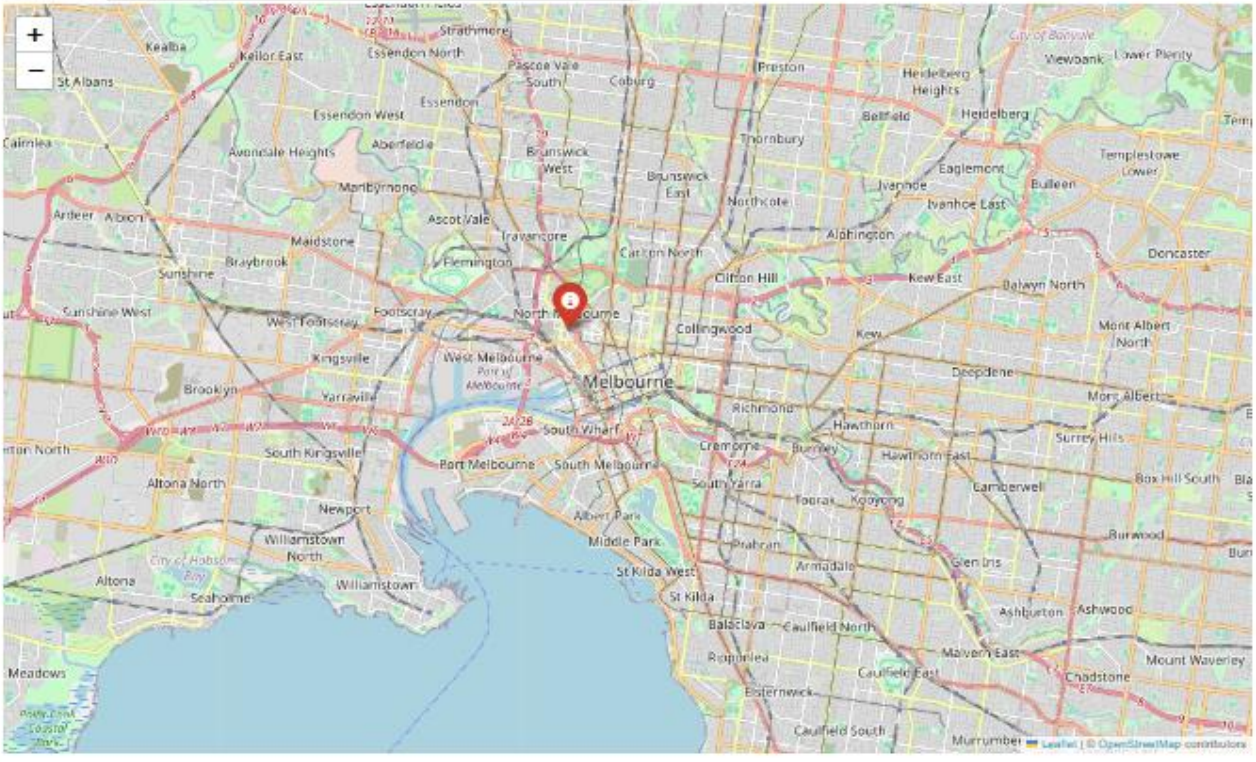
```
[ ]:
```

Customizing Markers

Markers can be customized with different icons and colours:

[14]:
```
from IPython.display import clear_output

def update_map(trading_name):
    clear_output(wait=True)

    # Create a fresh map
    m = folium.Map(location=[-37.8136, 144.9631], zoom_start=12)

    if trading_name == "All Locations":
        for _, row in df.iterrows():
            folium.Marker(
                location=[row['latitude'], row['longitude']],
                popup=row['trading_name'],
                icon=folium.Icon(color='red', icon='info-sign')
            ).add_to(m)
    else:
        selected_row = df[df['trading_name'] == trading_name].iloc[0]
        folium.Marker(
            location=[selected_row['latitude'], selected_row['longitude']],
            popup=selected_row['trading_name'],
            icon=folium.Icon(color='red', icon='info-sign')
        ).add_to(m)

    # Display the updated map
    display(m)

# Connect the dropdown to the update_map function
widgets.interact(update_map, trading_name=dropdown)
```

Select Place: North Star Hotel



[14]: <function __main__.update_map(trading_name)>

8. Conclusion

This guide has walked you through the process of creating an interactive map using folium and ‘ipywidgets’ in Jupyter Notebook. By integrating a dropdown filter, you can dynamically display and filter specific data points on your map, providing a powerful tool for data visualization and exploration. Whether you are a beginner or looking to enhance your mapping capabilities, these techniques allow you to create engaging and informative geographic visualizations that can be easily customized and shared. Continue experimenting with different datasets and customizations to further expand your skills and make your visualizations even more impactful.

Author

Venuka 2024.v1