Creating an Interactive Dashboard from Jupyter Notebook with Voila

Loading the Data

We're going to use the going to use the San Fransisco dataset (Available on Moodle). The dataset contains more than 150,000 rows of crime history that occurred in San Fransisco during 2016.

As usual, we can load the dataset with Pandas. To get a sense of what the dataset looks like, let's print out the first five rows of the dataset afterward.

Creating Interactive Widgets

Now that we've loaded the data, we can start right away to create widgets. These widgets are essentials to add interactivity to our visualizations.We're going to use three widgets: one slider widget and two multiple selection widgets. To create these widgets, we can use `ipywidgets` library that is available for Jupyter Notebook.

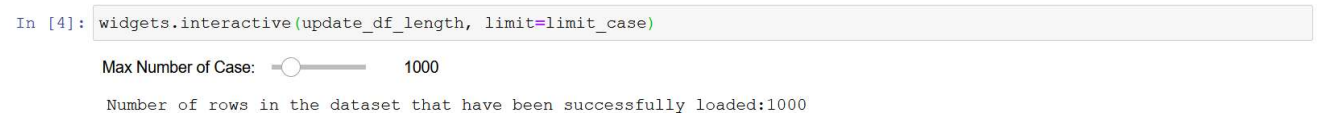
The first widget that we are going to create is the slider widget. To do this, we can use `IntSlider()` attribute from `ipywidgets`. This slider widget will control how many rows of the dataset that Pandas should load. For example, if the slider value is 1000, then Pandas should load the first 1000 rows of the dataset. Below is the code implementation of that.

```
import ipywidgets as widgets
import pandas as pd
style = {'description_width': 'initial'}limit_case = widgets.IntSlider(
    value=1000,
    min=100,
    max=5000,
    step=1,
    description='Max Number of Case:',
    disabled=False,
    style=style)
```

In the code above, we passed several arguments into `IntSlider()` attribute from `ipywidgets`. The first one is `value` , which is the default value that will be displayed when we run the code. Next, `min` and `max` are the minimum and maximum range values that can be considered in the slider. Meanwhile, `step` is the value increment or decrement when we move the slider up or down. Finally, we add `style` argument such that the word in `description` will not be truncated.

Next, with `interactive()` attribute from `ipywidgets`, we can link up the widget with the variable in which the value we want to interactively change. Now if we change the slider value, we can see that the length of the dataset will be changed accordingly.

If you run the code cells above, you'll get an interactive slider widget like this.



Next, let's create the second widget, which is the multiple selection widget. We can do this by using `SelectMultiple()` attribute from `ipywidgets`. With this widget, we have the option to visualize the crime only in particular selection of districts instead of all districts. In order to create this multiple selection widget, below is the code implementation.

```
import pandas as pd
import ipywidgets as widgets
from ipywidgets import Layout

df = pd.read_csv('SF_crimes.csv')
unique_district = df.PdDistrict.unique()
district = widgets.SelectMultiple(
    options = unique_district.tolist(),
    value = ['BAYVIEW', 'NORTHERN'],
    description='District',
    disabled=False,
    layout = Layout(width='50%', height='80px', display='flex')
)
```

In the code above, we use `SelectMultiple()` attribute to enable us to pick more than one values of our district variable. The first argument that we should specify is `options` , which should contain the list of available options of our variable (in our case different kinds of San Fransisco districts). The next one is `value` , which should contain the variable values that we want to display as default, and then `description` is for the text field to describe the name of the widget.

If you run the code cells above, you'll get the following interactive multiple selection widget.

To wrap up, we can create the third widget that is exactly the same as the previous multiple selection widget. The purpose of this widget is to enable us to choose which category of crimes that we want to visualize. Below is the code implementation of this widget.

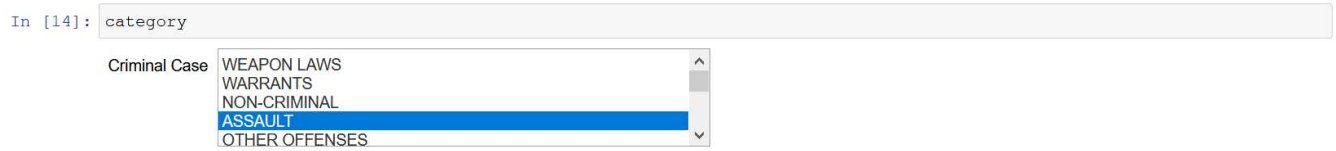
```
import pandas as pd
import ipywidgets as widgets
from ipywidgets import Layout

df = pd.read_csv('SF_crimes.csv')
unique_cat = df.Category.unique()style = {'description_width':
'initial'}category = widgets.SelectMultiple(
    options = unique_cat.tolist(),
    value = ['VANDALISM', 'ASSAULT', 'ROBBERY'],
    description='Criminal Case',
    disabled=False,
    style=style,
```

```
    layout = Layout(width='50%', height='80px')
)
```

The arguments that we passed on this `SelectMultiple()` attribute is the same as before, except that the `value` and `options` arguments would be the crime categories instead of districts.

If you run the code cells above, you'll get the following widget.



Next, we want to combine all of the three widgets that we've created before to create an interactive visualization. To visualize the crime locations, we can visualize them with a map, since we have the information regarding the latitude and longitude in the dataset.

To visualize the dataset into a map, we can use `folium` library. If you haven't installed `folium` yet, you can install it using pip command.

```
pip install folium
```

We can integrate the map with the widgets such that when we make a different selection with the widgets, the visualization will adjust accordingly.

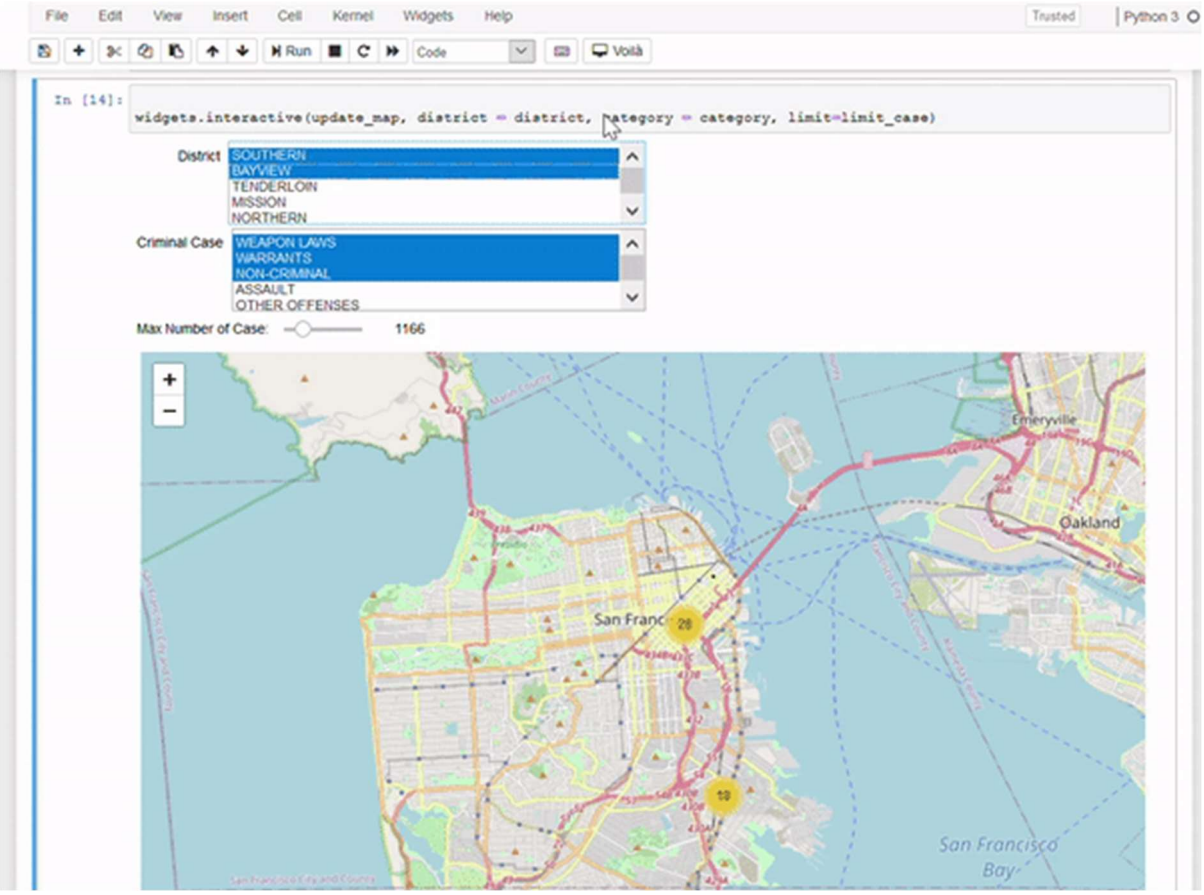
First, the value of the slider widget will determine how many crimes in the dataset that we should consider in the visualization. Next, the map should then show the crime categories and the districts based on our selection with the multiple selection widgets.

To create additional visualizations, we can also create two bar charts. One bar chart to show how many crimes are there based on the categories that we've selected using the widgets. The other chart is to show how many crimes are there in the district that we've selected using the widget.

Finally, we need to define a function to integrate all of the three widgets with our map visualization and the two bar charts.

Below is the complete code implementation to do all of that from start to finish.

Now if we call the function `update_map` in the last code cell of Jupyter Notebook above, we get the interactivity to our map and bar chart visualizations based on the value that we've selected in all of the three widgets.



Now we're done with our job with Jupyter Notebook to create interactive visualizations with the help of `ipywidgets` library!

Creating a Dashboard from Jupyter with Voila

So far, we've created interactive visualizations with the help of `ipywidgets` , `matplotlib` , and `folium` library. As of right now, you can proceed to show the visualization in your Jupyter Notebook to other people.

However, in most cases, you'll show the visualization to non-technical people, people who're not interested to see the chunks of Python code in your notebook and just want to see a clean visualization. Hence, using Jupyter Notebook to show the interactive visualization wouldn't be the best choice.

To transform the visualization on your Jupyter Notebook to a standalone dashboard, we can use Voila. Now if you haven't installed Voila yet, you can install it using pip command as follows:

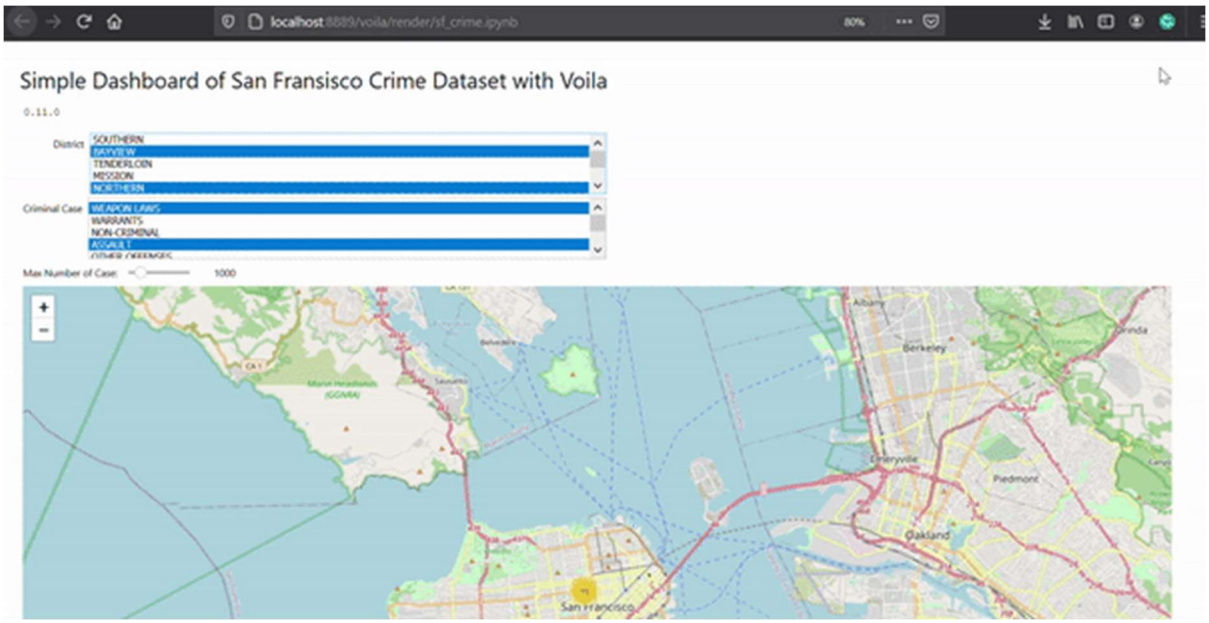
```
pip install voila
```

Once you've installed Voila, it's going to be very easy to create a standalone dashboard. All you need to do is go to your terminal or Anaconda prompt, and then type the following format:

```
voila path/to/your/notebook.ipynb
```

Alternatively, you can do so directly in your notebook by clicking the Voila icon in your Jupyter Notebook toolbar.

Now if you did either of two steps above, you'll get the following standalone dashboard.



And that's it! We have transformed the visualizations in our Jupyter Notebook into a standalone dashboard. Now with the same trick you can transform your own visualization with any dataset in Jupyter Notebook to a standalone dashboard.