schulzdLab4

January 12, 2021

1 Lab 4 - Data App

David Schulz

1.1 Introduction

As part of their roles, data scientists may create "data apps" that enable non-technical users like business analysists to analyze data on their own. Data apps use interactive visualizations and simple widgets to support basic queries and filtering. Dashboards are a special type of data app that sources data from a database or other live source enabling real-time monitoring of business data.

Bokeh (Python) is an open-source tool for creating data apps that are commonly used by data scientists. Commercially-supported dashboard tools include Tableau and Microsoft Power BI. Bokeh apps often generate single-page web apps saved as a HTML file. The logic for constructing the app is written in Python and converted to HTML and JavaScript by the Bokeh library.

In this lab, we are going to use Bokeh to create a data app to explore the real estate data we worked with previously.

1.2 Part 1: Display Real Estate on a Scatter Plot

[1]:
```
import pandas as pd
from bokeh.models import ColumnDataSource
from bokeh.plotting import Figure
from bokeh.io import show, output_notebook
from bokeh.models.tools import HoverTool

output_notebook()

def make_plot(cds:ColumnDataSource) -> Figure:
    plot = Figure(title=None, x_axis_label='Latitude', y_axis_label='Longitude',␣
 ↪plot_width=600, plot_height=500)
    plot.circle(x='latitude', y='longitude', source=cds, size=5, color='color')
    hover = HoverTool()
    hover.tooltips=[
```

```
        ('Address', '@street, @city, @state @zip'),
        ('Price', '@price'),
        ('Square Footage', '@sq__ft'),
        ('# of Beds', '@beds'),
        ('# of Baths', '@baths')
    ]
    plot.add_tools(hover)
    return plot

data = pd.read_csv('Sacramentorealestatetransactions.csv')
data['color'] = data['type'].map({'Residential':'red', 'Condo':'green',␣
 ↪'Multi-Family':'blue', 'Unkown':'orange'})
cds = ColumnDataSource(data=data)
show(make_plot(cds))
```

1.3 Part 2: Refine ColumnDataSource Object Based on Search Criteria

[2]:
```
from pandas import DataFrame

def make_dataset(df:DataFrame, type_range, price_range, baths_range, beds_range,␣
 ↪sqft_range) -> ColumnDataSource:
    df = df[df['type'].isin(type_range)]
    df = df[(df['price'] >= price_range[0]) & (df['price'] <= price_range[1])]
    df = df[(df['baths'] >= baths_range[0]) & (df['baths'] <= baths_range[1])]
    df = df[(df['beds'] >= beds_range[0]) & (df['beds'] <= beds_range[1])]
    df = df[(df['sq__ft'] >= sqft_range[0]) & (df['sq__ft'] <= sqft_range[1])]
    return ColumnDataSource(data=df)

cds = make_dataset(data, ['Residential'], [50000, 75000], [1, 2], [1, 2], [1000,␣
 ↪2000])
show(make_plot(cds))
```

1.4 Part 3: Add Widgets and Create an Interactive Visualization

[3]:
```
from bokeh.models.widgets import CheckboxGroup, RangeSlider
from bokeh.models import Column
from bokeh.layouts import column, row

housing_selection = CheckboxGroup(labels=data['type'].unique().tolist(),␣
 ↪active=[0, 1, 2, 3])
range_slider_price = RangeSlider(start=0, end=900000, value=(0,900000), step=1,␣
 ↪title='Price')
```

```
range_slider_baths = RangeSlider(start=0, end=5, value=(0,5), step=1,␣
 ↪title='Baths')
range_slider_beds = RangeSlider(start=0, end=8, value=(0,8), step=1,␣
 ↪title='Beds')
range_slider_sq_ft = RangeSlider(start=0, end=6000, value=(0,6000), step=1,␣
 ↪title='Square Footage')
controls = Column(housing_selection, range_slider_price, range_slider_baths,␣
 ↪range_slider_beds, range_slider_sq_ft)

source = ColumnDataSource(data=data)
p = make_plot(source)

# Update function takes three default parameters
def update(attr, old, new):
    # Get the list of carriers for the graph
    selected_type = [housing_selection.labels[i] for i in housing_selection.
 ↪active]
    price_range = [range_slider_price.value[0], range_slider_price.value[1]]
    baths_range = [range_slider_baths.value[0], range_slider_baths.value[1]]
    beds_range = [range_slider_beds.value[0], range_slider_beds.value[1]]
    sq_ft_range = [range_slider_sq_ft.value[0], range_slider_sq_ft.value[1]]

    # Make a new dataset based on the selected carriers and the
    # make_dataset function defined earlier
    new_src = make_dataset(data, selected_type, price_range, baths_range,␣
 ↪beds_range, sq_ft_range)
    # Update the source used in the quad glpyhs
    source.data.update(new_src.data)

def modify_doc(doc):
    housing_selection.on_change('active', update)
    range_slider_price.on_change('value', update)
    range_slider_baths.on_change('value', update)
    range_slider_beds.on_change('value', update)
    range_slider_sq_ft.on_change('value', update)

    doc.add_root(row(p,column(controls)))

show(modify_doc)
```