Lab 04: Data App
CS3300 Data Science

Learning Outcomes

1. Understand the basic process of data science and exploratory data analysis including modes of inquiry (hypothesis driven, data driven, and methods driven).
2. Identify, access, load, and prepare (clean) a data set for a given problem.
3. Communicate findings through generated data visualizations and reports.

Overview

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, you are going to use Bokeh to create a data app to explore the real estate data you worked with previously.

Bokeh's documentation:
https://bokeh.pydata.org/en/latest/index.html

Helpful Bokeh Dashboard Tutorials:
Part I - https://towardsdatascience.com/data-visualization-with-bokeh-in-python-part-one-getting-started-a11655a467d4
Part II - https://towardsdatascience.com/data-visualization-with-bokeh-in-python-part-ii-interactions-a4cf994e2512
Part III - https://towardsdatascience.com/data-visualization-with-bokeh-in-python-part-iii-a-complete-dashboard-dc6a86aa6e23

Instructions

You should use the Bokeh library to create a dashboard to visualize and explore the given dataset. The dashboard/visualization should run completely in your Jupyter notebook. For this submission you will need to submit the .ipynb file that has a working version of the visualization and a pdf of the finished notebook. It should be noted that Bokeh supports development of apps/dashboards as Python scripts. With this approach you can use the library to create a single page web app (HTML file). If you wish to explore this functionality in this lab, you have the freedom – but the submission should still be in the form of a Jupyter notebook.

Part 1: Display Real Estate on a Scatter Plot

Define a method (make_plot) that accepts a ColumnDataSource (bokeh.models) as an argument. This method should return a bokeh figure object (bokeh.plotting). You may use show() to display this figure. You will want to add a "color" feature to your dataframe that specifies a unique hex color that matches the residence type.

- Data Frames can be used in the creation of ColumnDataSource objects (example)
- Use the latitudes and longitudes to create a scatter plot (example). Label the axes.
- Color the points by residence type. You may wish to use color pallets from bokeh.palettes.
- When the user puts their mouse over a point, display the address, price, square footage, number of beds, and number of baths as a tooltip. (figure.add_tools(Hovertool))

At the end of experiment 1, you should have a make_plot method and display the whole dataset using show(figure).

Part 2: Refine ColumnDataSource Object based on Search Criteria

Now that you have a method definition that creates a scatterplot from a ColumnDataSource, you will want to be able to create new plots depending on search criteria. Create a new method definition (make_dataset) that accepts a dataframe and separate lists that describe ranges of values for each of the features you are interested in. This method should return a new ColumnDataSource object.

- Your method should accept lists describing the following inclusion criteria for the ColumnDataSource that you will create.
    - Residential type
    - Price range
    - Baths range
    - Beds range
    - Square Foot range
- Your method should "filter" your given dataframe to only include entries that meet the range/search criteria.

At the end of experiment 2, you should have a make_dataset method and display a plot of all the residential properties valued between $50,000 and $75,000 with 1-2 bathrooms, between 1000 to 2000 square feet, and 1-2 bedrooms.

Part 3: Add Widgets and Create and Interactive Visualization

With ability to create a new ColumnDataSource given ranges/filter parameters, you can now incorporate controls for your end user to specify desired ranges. Create widgets to add to your visualization/data app to set the desired filter criteria.

- CheckboxGroup – For residential type
- RangeSlider – For Price, Square Footage, Number of Beds, and Number of Baths.

To help ease event handling in python/bokeh, we have provided the methods at the end of the document that should be placed after the widget definitions. When run, this should display your working figure. (Note: you will want to design your widgets, columns, and controls to adhere to the variable names/definitions provided here).

- Check the code appendix at the end of this prompt for event handler definitions and helpful import statements. These import statements are meant to help guide you through the Bokeh library used in this lab.
- After defining the widgets and grouping them in a Column object (Bokeh.models.Column) you will have to initialize a source (using make_dataset) and figure_object (using make_plot).
- Bonus Material – It is often helpful to display tabular data based on the returned observations. You may choose to implement a table widget that is refined based on search criteria. Try to display the address and associated price.

This experiment should conclude with the interactive visualization that includes working sliders for Price, Square Footage, Number of Beds, and Number of Baths.

Submission Instructions

You will submit two separate files to Canvas. The .ipynb file containing your experiments and working visualization app. A pdf copy of your notebook.

Rubric

Followed submission instructions	5%
Formatting: Report is polished and clean. No unnecessary code. Section headers are used. Plots are described and interpreted using text. The report contains an introduction and conclusion.	5%
Part 1: Display Real Estate on a Scatter Plot	
Scatter plot of longitude and latitude	10%
Tool tip functions appropriately	10%
Use of color pallet to distinguish residential type	5%
Function (make_plot) meets specification	5%
Part 2: Refine ColumnDataSource Object based on Search Criteria	
Function narrows/filters dataset based on given parameters	15%
Function (make_dataset) meets specification	5%
Expected plot (shows filtered observations)	5%
Part 3: Add Widgets and Create and Interactive Visualization	
Widget Creation	15%
Functional Data App	10%
Technical/Organizational/Presentation Proficiency	10%

Code Appendix

Import Statements

*This should be copied into the first cell of the notebook

```
import pandas as pd
import numpy as np

from bokeh.io import show, output_notebook, push_notebook, output_file

from bokeh.plotting import figure
from bokeh.models import ColumnDataSource, HoverTool, Column
from bokeh.palettes import all_palettes

from bokeh.models.widgets import CheckboxGroup, RangeSlider, DataTable,
DateFormatter, TableColumn
from bokeh.layouts import column, row, WidgetBox
from bokeh.application.handlers import FunctionHandler
from bokeh.application import Application

output_notebook()
```

Event Handlers and Figure Display Statements

*This should be copied into the final cell of the notebook

```
# 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]]
    sq_ft_range = [range_slider_sq_ft.value[0], range_slider_sq_ft.value[1]]
    beds_range = [range_slider_beds.value[0], range_slider_beds.value[1]]

    # Make a new dataset based on the selected carriers and the
    # make_dataset function defined earlier
    new_src = make_dataset(df_exp3, selected_type, price_range, baths_range,
sq_ft_range, beds_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(figure_object,column(controls)))
    #If you want to add A table to the visualization
    #doc.add_root(row(figure_object,column(controls)))

show(modify_doc)
```