CS 3300 Data Science - Lab 4: Data App

Stuart Harley

Introduction

The most sophisticated statistical analysis can be meaningless without an effective means for communicating the results. Results will have little impact if they cannot be clearly communicated, and often the best way for presenting the results of an analysis is with visualizations. Visualizations are also more effective if they are interactive. This allows people to play with the data to see how changing parameters affects the results.

One such interactive visualization library in Python is Bokeh. In this lab, we use Bokeh to visualize our San Francisco Real Estate Data that we have been exploring for the past several labs. At the end of this lab, we create an interactive plot that shows the different real estate values on a 2d plot of Latitude and Longitude, with widgets that allow you to change parameters to limit which points show up on the plot.

Importing Libraries

In [1]:
```
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,
DateFo rmatter, TableColumn
from bokeh.layouts import column, row, WidgetBox
from bokeh.application.handlers import FunctionHandler
from bokeh.application import Application

output_notebook()
```

(https://bokehBokehJS.org)2.1.1 successfully loaded.

Importing Data

In [2]:
```
df = pd.read_csv('CleanedSacramentorealestatetransactions.csv', \
                 dtype={'city': 'category', 'zip': 'category', \
                        'state': 'category', 'beds': 'category', \
                        'baths': 'category', 'type': 'category', \
                        'street_type': 'category'})
```

Adding a color feature that specifies a unique hex color that matches the residence type.

In [3]:
```
color_map = {'Condo': '#ff0000', 'Multi-Family': '#0000ff', 'Residential': '#0
0ff00'}
df['color'] = df['type'].map(color_map)
```

In [4]:
```
df.head()
```

Out[4]:

	street	city	zip	state	beds	baths	sq__ft	type	sale_date	price
0	3526 HIGH ST	SACRAMENTO	95838	CA	2	1	836	Residential	Wed May 21 00:00:00 EDT 2008	59222
1	51 OMAHA CT	SACRAMENTO	95823	CA	3	1	1167	Residential	Wed May 21 00:00:00 EDT 2008	68212
2	2796 BRANCH ST	SACRAMENTO	95815	CA	2	1	796	Residential	Wed May 21 00:00:00 EDT 2008	68880
3	2805 JANETTE WAY	SACRAMENTO	95815	CA	2	1	852	Residential	Wed May 21 00:00:00 EDT 2008	69307
4	6001 MCMAHON DR	SACRAMENTO	95824	CA	2	1	797	Residential	Wed May 21 00:00:00 EDT 2008	81900

In [5]:
```
df.info()
```

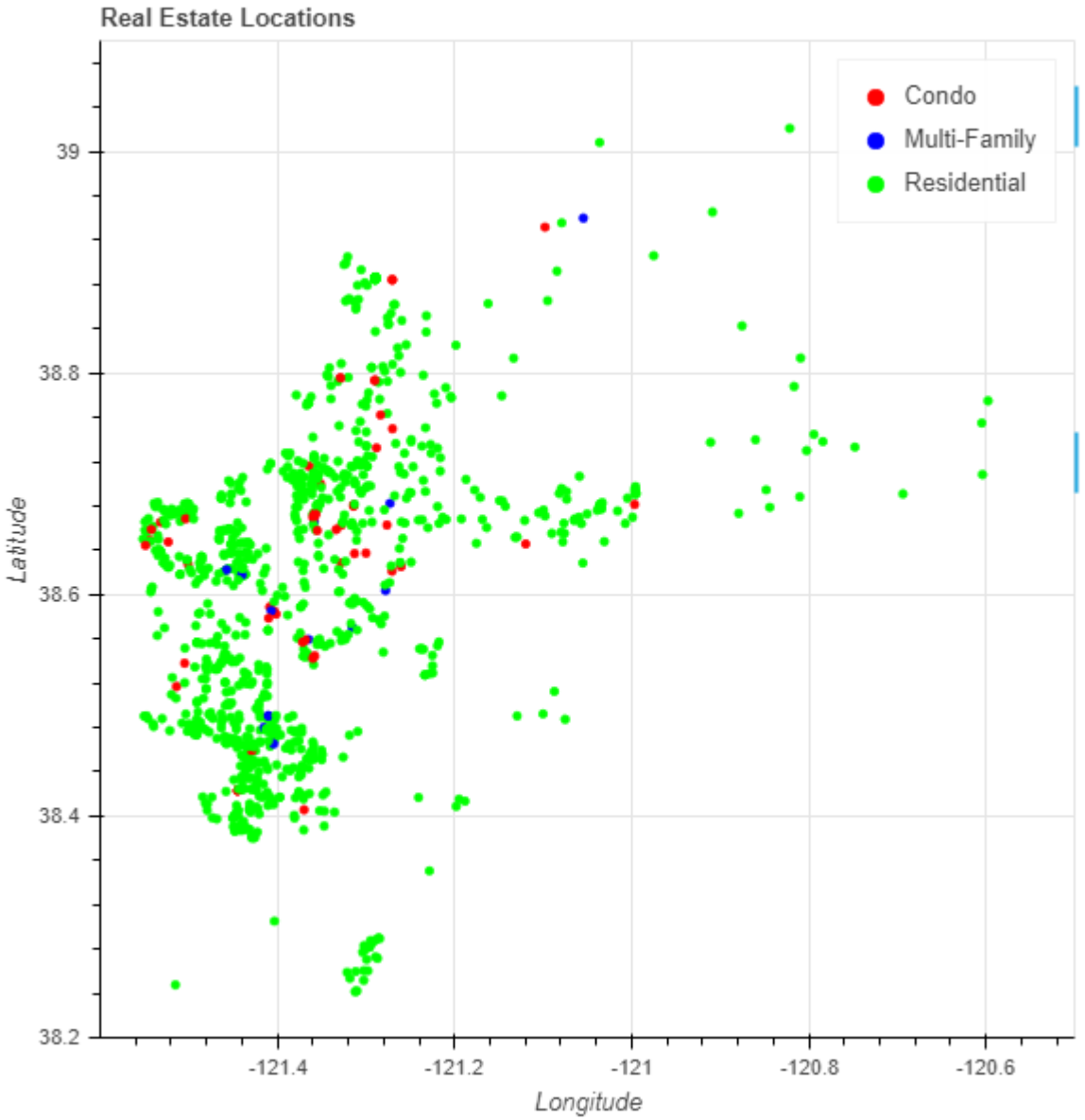
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 984 entries, 0 to 983
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   street       984 non-null    object
 1   city         984 non-null    category
 2   zip          984 non-null    category
 3   state        984 non-null    category
 4   beds         984 non-null    category
 5   baths        984 non-null    category
 6   sq__ft       984 non-null    int64
 7   type         984 non-null    category
 8   sale_date    984 non-null    object
 9   price        984 non-null    int64
 10  latitude     984 non-null    float64
 11  longitude    984 non-null    float64
 12  empty_lot    984 non-null    bool
 13  street_type  984 non-null    category
 14  color        984 non-null    category
dtypes: bool(1), category(8), float64(2), int64(2), object(2)
memory usage: 61.1+ KB
```

Part 1: Display Real Estate on a Scatter Plot

Defining a make_plot method. This method creates a bokeh figure that shows a scatter plot of the properties from the data based on their latitudinal and longitudinal locations. The color of the points corresponds to their property "type" and there is a hovertool included which shows additional data about the property when hovering over the point.

In [6]:
```
def make_plot(cds):
    fig = figure(title='Real Estate Locations', x_axis_label='Longitude',
                 y_axis_label='Latitude', x_range=(-121.6, -120.5), y_range=(38.
2, 39.1))
    fig.scatter(x='longitude', y='latitude', color='color',
legend_group='typ e', source=cds)
    fig.add_tools(HoverTool(tooltips=[('Address', '@street @city, @state @zip'
),
                                      ('Price', '$@price'), ('Sq Ft', '@sq__f
t'),
                                       ('Beds', '@beds'), ('Baths', '@baths'
)]))
    return fig
```

In [7]:
```
cds = ColumnDataSource(df)
fig = make_plot(cds)
show(fig)
```



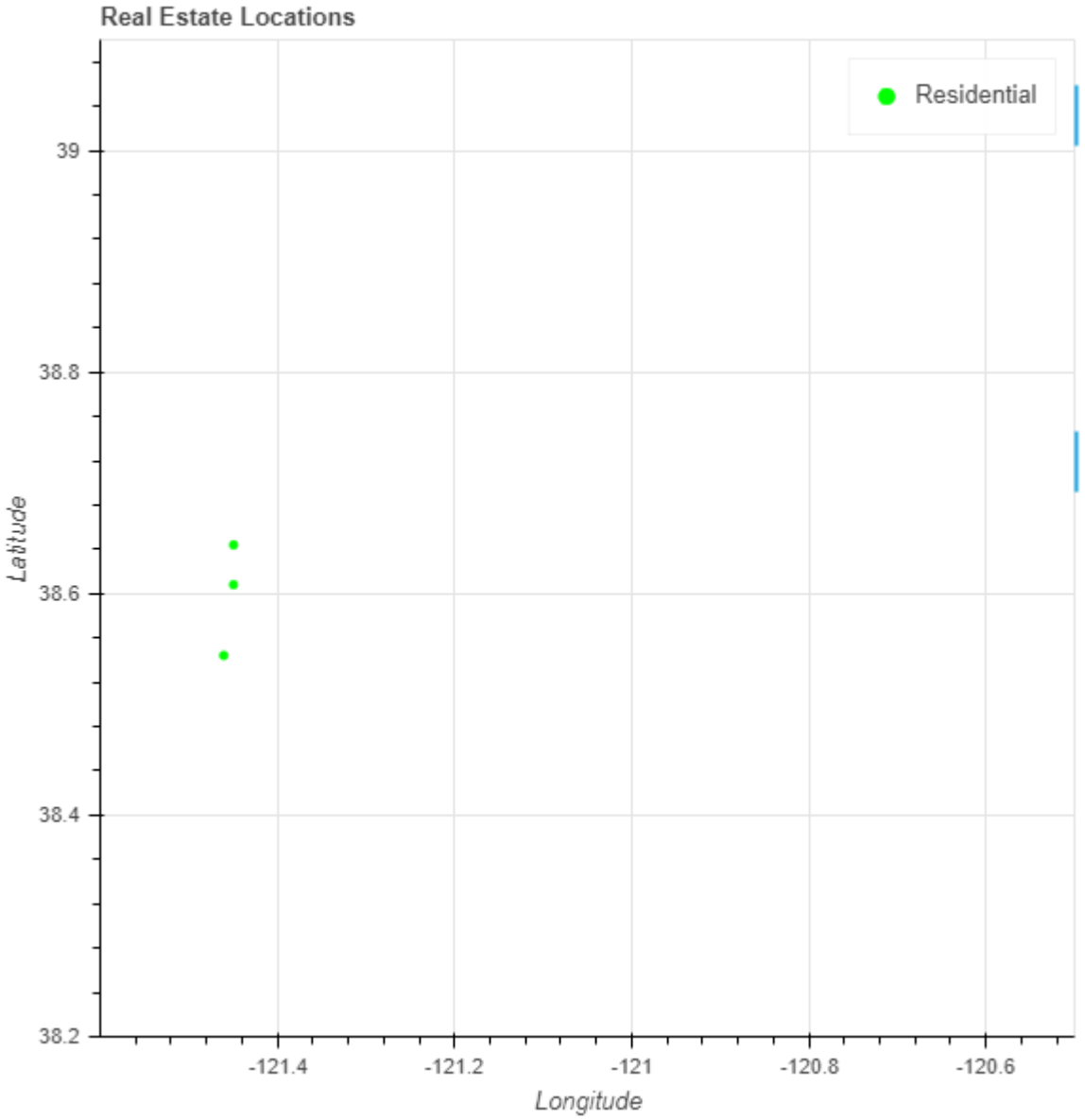
Part 2: Refine ColumnDataSource Object based on Search Criteria

Defining a make_dataset function. This function accepts a dataframe and separate lists that describe ranges of values for each of the features you are interested in. This method returns a new ColumnDataSource object.

In [8]:
```
def make_dataset(df, selected_type, price_range, baths_range, sq_ft_range, bed
s_range):
    temp_df = df[df.type.isin(selected_type)]
    temp_df = temp_df[temp_df.price.between(price_range[0], price_range[1])]
    temp_df = temp_df[temp_df.baths.isin(map(str, list(range(baths_range[0], b
aths_range[1]+1))))]
    temp_df = temp_df[temp_df.beds.isin(map(str, list(range(beds_range[0],
bed s_range[1]+1))))]
    temp_df = temp_df[temp_df.sq__ft.between(sq_ft_range[0],
    sq_ft_range[1])] return ColumnDataSource(temp_df)
```

Displaying a plot of all the residential properties valued between $50,000 and \$75,000 with 1-2 bathrooms, 1-2 bedrooms, and between 1000 to 2000 square feet.

In [9]:
```
selected_type = ['Residential']
price_range = [50000, 75000]
baths_range = [1,2]
beds_range = [1,2]
sq_ft_range = [1000, 2000]
cds = make_dataset(df, selected_type, price_range, baths_range, sq_ft_range,
b eds_range)
fig = make_plot(cds)
show(fig)
```



Part 3: Add Widgets and Create an Interactive Visualization

In [10]:
```
housing_selection = CheckboxGroup(labels=['Residential', 'Condo', 'Multi-Famil
y'], active=[0,1,2])
selected_type = [housing_selection.labels[i] for i in
housing_selection.active ]
range_slider_price = RangeSlider(start=0, end=df['price'].max(), value=(0,
df[ 'price'].max()), step=1000, title='Price Range')
price_range = [range_slider_price.value[0], range_slider_price.value[1]]
range_slider_baths = RangeSlider(start=0, end=5, value=(0,5), step=1,
title='B aths Range')
baths_range = [range_slider_baths.value[0], range_slider_baths.value[1]]
range_slider_sq_ft = RangeSlider(start=0, end=df['sq__ft'].max(), value=(0,
df ['sq__ft'].max()), step=100, title='Sq Ft Range')
sq_ft_range = [range_slider_sq_ft.value[0], range_slider_sq_ft.value[1]]
range_slider_beds = RangeSlider(start=0, end=8, value=(0,8), step=1,
title='Be ds Range')
beds_range = [range_slider_beds.value[0], range_slider_beds.value[1]]

controls = Column(housing_selection, range_slider_price, range_slider_baths,
r ange_slider_sq_ft, range_slider_beds)
source = make_dataset(df, selected_type, price_range, baths_range,
sq_ft_range , beds_range)
p = make_plot(source)
```

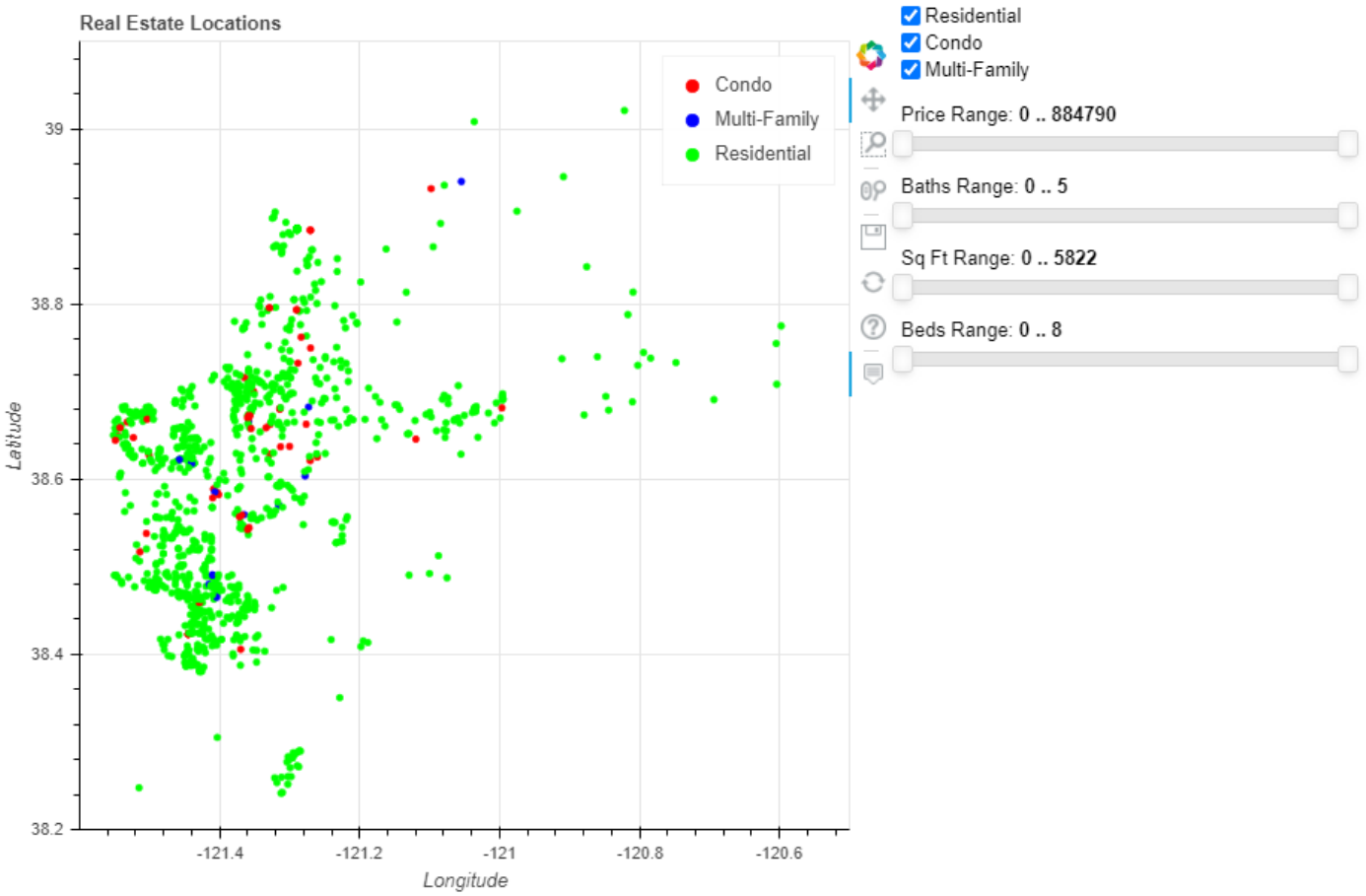
In [11]:
```
# Update function takes three default parameters
def update(attr, old, new):
    # Get the list of carriers for the graph
    selected_type = [housing_selection.labels[i] for i in
housing_selection.ac tive]
    price_range  =  [range_slider_price.value[0],  range_slider_price.value[1]]
    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, 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(p,column(controls)))
    #If you want to add A table to the visualization
    #doc.add_root(row(p,column(controls)))

show(modify_doc)
```



Conclusion

Bokeh is a good tool that can be used to make effective interactive visualizations of data. However, we did run into issues in getting the final visualization to display within the Jupyter Notebook running on Rosie. I was however, able to get the same notebook to run correctly within Anaconda's Jupyter Notebook. That aside, the visualization we created directly above is a good tool that could be used by a real estate company to show all the locations of their properties on a plot within a customers' wants. It is easily manipulated and understandable and shows a good representation of the data.