Lab 04 - Data App

Paige Rosynek, 01.10.2023

Introduction

In this lab, we will be creating a data app. In general, data apps are used to allow non-technical users to analyze the data through interactive visualizations on their own. For this lab, we will be using the Sacremento real estate data set along with the open-source Python library, Bokeh, to create interactive data visualizations for our data app. The goal of this lab is to create a dashboard that users can interact with to explore and visualize the real estate data under certain conditions specified by the user.

Import libraries

In [1]:
```
from bokeh.plotting import figure, show
from bokeh.models import ColumnDataSource, CheckboxGroup, CustomJS, RangeSlider, Column
from bokeh.io import output_notebook, reset_output, show, push_notebook
from bokeh.transform import factor_cmap
from bokeh.palettes import all_palettes
from bokeh.layouts import column, row
import pandas as pd
output_notebook()
```

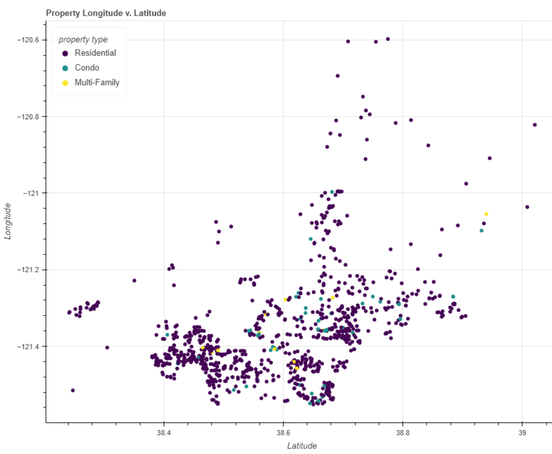

Part 1: Display Real Estate on a Scatter Plot

Import Sacremento data set

In [3]:
```
df = pd.read_csv('../data/sacramento.csv')
source = ColumnDataSource(data=df)
```

In [4]:
```
def make_plot(data):
    property_types = pd.unique(data.data['type'])

    TOOLTIPS = [
        ('address', '@street'),
        ('price', '$price'),
        ('sq_ft', '@sq__ft'),
        ('beds', '@beds'),
        ('baths', '@baths')
    ]

    fig = figure(width=900, height=700, tooltips=TOOLTIPS, title='Property Longitude v. Latitude',
                 x_axis_label='Latitude', y_axis_label='Longitude')

    fig.circle(x='latitude', y='longitude', source=data, size=5, legend_field='type',
               color=factor_cmap('type', palette='Viridis3', factors=property_types))

    fig.legend.title = 'property type'
    fig.legend.location = 'top_left'
    return fig
```

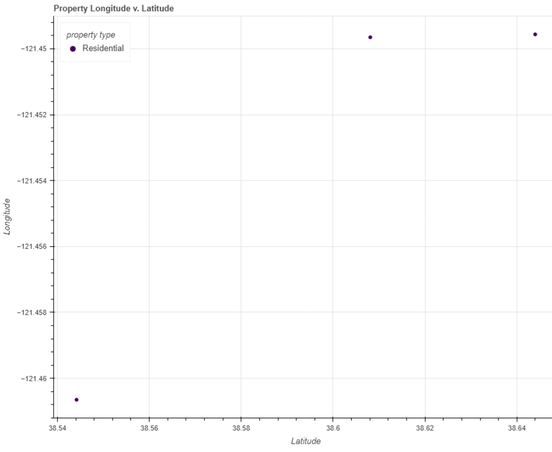
In [5]:
```
fig = make_plot(source)
show(fig)
```



Part 2: Refine ColumnDataSource Object based on Search Criteria

In [6]:
```
def make_dataset(df, types, price_range, baths_range, beds_range, sq_ft_range):
    new_df = df.copy(deep=True)
    new_df = new_df[new_df['type'].isin(types)]
    new_df = new_df[new_df['price'].between(price_range[0], price_range[1])]
    new_df = new_df[new_df['baths'].between(baths_range[0], baths_range[1])]
    new_df = new_df[new_df['beds'].between(beds_range[0], beds_range[1])]
    new_df = new_df[new_df['sq__ft'].between(sq_ft_range[0], sq_ft_range[1])]
    return ColumnDataSource(new_df)
```

In [7]:
```
filtered = make_dataset(df=df,
                        types=['Condo', 'Multi-Family', 'Residential'],
                        price_range=(50000, 75000),
                        baths_range=(1, 2),
                        beds_range=(1, 2),
                        sq_ft_range=(1000, 2000))
show(make_plot(filtered))
```



Part 3: Add Widgets and Create and Interactive Visualization

In [8]:
```
def modify_doc(doc):
    df = pd.read_csv('../data/sacramento.csv')

    # Instantiate here the CheckboxGroup and RangeSlider objects
    # Check the update method below to make sure you choose the same identifiers for the objects
    property_types = ['Condo', 'Multi-Family', 'Residential']
    housing_checkbox_group = CheckboxGroup(labels=property_types, active=[0,1,2])

    slider_price = RangeSlider(title='property price', start=0, end=df['price'].max(), step=1, value=(0, df['pr
    slider_beds = RangeSlider(title='number of beds', start=0, end=df['beds'].max(), step=1, value=(0, df['beds
    slider_baths = RangeSlider(title='number of baths', start=0, end=df['baths'].max(), step=1, value=(0, df['b
    slider_sqft = RangeSlider(title='sq_ft', start=0, end=df['sq__ft'].max(), step=1, value=(0, df['sq__ft'].ma

    # create the data source by calling the method make_dataset
    selected_types = [housing_checkbox_group.labels[i] for i in housing_checkbox_group.active]
    source = make_dataset(df=df,
                          types=selected_types,
                          price_range=[slider_price.value[0], slider_price.value[1]],
                          baths_range=[slider_baths.value[0], slider_baths.value[1]],
                          beds_range=[slider_beds.value[0], slider_beds.value[1]],
                          sq_ft_range=[slider_sqft.value[0],slider_sqft.value[1]]
                          )

    # call the method make_plot
    figure_object =  make_plot(source)

    # Update function takes three default parameters
    def update(attr, old, new):
        # Get the list of selected types
        selected_types = [housing_checkbox_group.labels[i] for i in housing_checkbox_group.active]

        # Make a new column source according to the selected properties
        source2 = make_dataset(df=df,
                               types=selected_types,
                               price_range=[slider_price.value[0], slider_price.value[1]],
                               baths_range=[slider_baths.value[0], slider_baths.value[1]],
                               beds_range=[slider_beds.value[0], slider_beds.value[1]],
                               sq_ft_range=[slider_sqft.value[0], slider_sqft.value[1]]
                              )

        # Update the data of the main source
        source.data.update(source2.data)


    housing_checkbox_group.on_change('active', update)
    slider_price.on_change('value', update)
    slider_beds.on_change('value', update)
    slider_baths.on_change('value', update)
    slider_sqft.on_change('value', update)

    controls = column(housing_checkbox_group, slider_price, slider_baths, slider_sqft, slider_beds) #pass in to

    doc.add_root(row(figure_object, controls))

show(modify_doc)
```

```
ERROR:bokeh.server.views.ws:Refusing websocket connection from Origin 'file://'; use --al
low-websocket-origin= or set BOKEH_ALLOW_WS_ORIGIN= to permit this; currently we allow origins {'localhost:888
8'}
WARNING:tornado.access:403 GET /ws (::1) 12.00ms
```

Conclusion

In this lab, we created a dashboard to display the Sacremento real estate data using the data visualization library, Bokeh. The final dashboard allows users to interact with a scatter plot of the data, plotted as Longitude v. Latitude. The data visualization dynamically changes according to the filters selected by the user. The filters for the final dashboard include: property type, price range, number of baths range, number of beds range, and square footage range. Overall, I learned that Bokeh is a useful library for creating interactive data visualizations for data apps such that people without technical knowledge of Python can interact with, and analyze the data.