dashboard

April 10, 2021

1 San Francisco Rental Prices Dashboard

In this notebook, you will compile the visualizations from the previous analysis into functions that can be used for a Panel dashboard.

[1]:
```
# imports
import panel as pn
import plotly.express as px
import pandas as pd
import matplotlib.pyplot as plt
import os
from pathlib import Path
from dotenv import load_dotenv
```

[2]:
```
# Set up Panel Plotly extension
pn.extension('plotly')
```

[3]:
```
# Import hvplot.pandas after pn.extension
# This avoids plotly initialization failure
import hvplot.pandas
```

[4]:
```
# Read the Mapbox API key
load_dotenv('api_keys.env')
map_box_api = os.getenv("MAPBOX_TOKEN")
px.set_mapbox_access_token(map_box_api)
```

2 Import Data

[5]:
```
# Import the necessary CSVs to Pandas DataFrames
file_path_census = Path("Data/sfo_neighborhoods_census_data.csv")
df_costs = pd.read_csv(file_path_census, index_col="year")

file_path_coord = Path("Data/neighborhoods_coordinates.csv")
neighborhood_coordinates = pd.read_csv(file_path_coord)
neighborhood_coordinates.columns=["neighborhood", "lat", "log"]

df_costs.dropna(inplace=True)
```

2.1 Panel Visualizations

In this section, you will copy the code for each plot type from your analysis notebook and place it into separate functions that Panel can use to create panes for the dashboard.

These functions will convert the plot object to a Panel pane.

Be sure to include any DataFrame transformation/manipulation code required along with the plotting code.

Return a Panel pane object from each function that can be used to build the dashboard.

Note: Remove any `.show()` lines from the code. We want to return the plots instead of showing them. The Panel dashboard will then display the plots.

[6]:
```
# Define Panel Visualization Functions
def housing_units_per_year():
    housing_units_per_year = df_costs.groupby("year")["housing_units"].mean()
    bar_min = housing_units_per_year.describe(include='all').loc['min'] -␣
↪(housing_units_per_year.describe(include='all').loc['std']/4)
    bar_max = housing_units_per_year.describe(include='all').loc['max'] +␣
↪(housing_units_per_year.describe(include='all').loc['std']/4)

    housing_units_per_year_plot = housing_units_per_year.hvplot.
↪bar(xlabel='Year',
                                                                  ␣
↪ylabel='Housing Units',
                                                                  ␣
↪ylim=(bar_min, bar_max),
                                                                   height=400).
↪opts(title='Housing Units in San Francisco from 2010 to 2016',
                                                                             ␣
↪     yformatter="%.0f")
    return housing_units_per_year_plot

def average_gross_rent():
    average_gross_rent = df_costs.groupby(['year']).mean()
    average_gross_rent_plot = average_gross_rent['gross_rent'].hvplot.
↪line(line_color='red',
                                                          xlabel='Year',
                                                          ylabel='Price␣
↪per SqFt',
                                                          width=500,
                                                          height=400,
                                                          grid=True,
                                                          title='Average␣
↪Gross Rent by Year'
```

```
                                                                    )
    return average_gross_rent_plot

def average_sales_price():
    average_sales_price = df_costs.groupby(['year']).mean()
    average_sales_price_plot = average_sales_price['sale_price_sqr_foot'].
 ↪hvplot.line(line_color='purple',
                                                                        ␣
 ↪xlabel='Year',
                                                                        ␣
 ↪ylabel='USD',
                                                                         width=500,
                                                                         height=400,
                                                                         grid=True,
                                                                         title =␣
 ↪'Average Price per SqFt by Year')
    return average_sales_price_plot

def average_price_by_neighborhood():
    average_price_by_neighborhood = df_costs.groupby(['year', 'neighborhood']).
 ↪mean()
    neighborgood = average_price_by_neighborhood['sale_price_sqr_foot']
    average_price_by_neighborhood_plot = neighborgood.
 ↪hvplot(groupby='neighborhood',
                                                          line_color='blue',
                                                          xlabel='Year',
                                                          ylabel='Avg. Sale␣
 ↪Price per Square Foot',
                                                          width=600,
                                                          height=300,
                                                          grid=True
                                                         )
    return average_price_by_neighborhood_plot

def top_most_expensive_neighborhoods():
    top_most_expensive_neighborhoods = df_costs.groupby(['neighborhood']).mean()
    top_most_expensive_neighborhoods.sort_values('sale_price_sqr_foot',␣
 ↪ascending=False, inplace=True)
    top_most_expensive_neighborhoods = top_most_expensive_neighborhoods[:10]
    top_most_expensive_neighborhoods_plot = top_most_expensive_neighborhoods.
 ↪hvplot.bar(height=400,
                                                                        ␣
 ↪        x='neighborhood',
                                                                        ␣
 ↪        xlabel='Neighborhood',
```

```
        y='sale_price_sqr_foot',
        ylabel='Avg. Sale Price per Square Foot',
        rot=90).opts(title='Top 10 Expensive Neighborhoods in SFO')
    return top_most_expensive_neighborhoods_plot

def most_expensive_neighborhoods_rent_sales():
    sfo_neighborhood_avg = df_costs.groupby(['year', 'neighborhood']).mean()
    most_expensive_neighborhoods_rent_sales_plot = sfo_neighborhood_avg.hvplot.bar(groupby='neighborhood',
    height=400,
    x='year',
    xlabel='Year',
    y=['sale_price_sqr_foot','gross_rent'],
    ylabel='Price',
    rot=90,
    title='Comparing Cost to Purchase Versus Rental Income')
    return most_expensive_neighborhoods_rent_sales_plot


def parallel_coordinates():
    sfo_top_neighborhood = df_costs.groupby(['neighborhood']).mean()
    sfo_top_neighborhood.sort_values('sale_price_sqr_foot', ascending=False,inplace=True)
    df_expensive_neighborhoods = sfo_top_neighborhood[:10]
    df_expensive_neighborhoods.reset_index(inplace=True)
    parallel_coordinates_plot = px.parallel_coordinates(df_expensive_neighborhoods, color='sale_price_sqr_foot')
    return parallel_coordinates_plot


def parallel_categories():
    sfo_top_neighborhood = df_costs.groupby(['neighborhood']).mean()
    sfo_top_neighborhood.sort_values('sale_price_sqr_foot', ascending=False,inplace=True)
    df_expensive_neighborhoods = sfo_top_neighborhood[:10]
    df_expensive_neighborhoods.reset_index(inplace=True)
```

```
    parallel_categories_plot = px.parallel_categories(
        df_expensive_neighborhoods,
        dimensions=['neighborhood','sale_price_sqr_foot','housing_units',␣
↪'gross_rent'],
        color='sale_price_sqr_foot',
        color_continuous_scale=px.colors.sequential.Inferno
    )
    return parallel_categories_plot

def neighborhood_map():

    neighborhood_avg = df_costs.groupby('neighborhood').mean()
    neighborhood_avg.reset_index(inplace=True)
    neighborhood_map = pd.merge(neighborhood_coordinates, neighborhood_avg,␣
↪on='neighborhood')
    neighborhood_plot = px.scatter_mapbox(
        neighborhood_map,
        lat="lat",
        lon="log",
        size="sale_price_sqr_foot",
        color="gross_rent",
        zoom=11,
        color_continuous_scale=px.colors.cyclical.IceFire,
        size_max=15,
        title='Average Sale Price Per Square Good and Gross Rent in San␣
↪Francisco'
    )
    return neighborhood_plot

def sunburst():
    sfo_top_neighborhood = df_costs.groupby(['neighborhood']).mean()
    sfo_top_neighborhood.sort_values('sale_price_sqr_foot', ascending=False,␣
↪inplace=True)
    df_expensive_neighborhoods = sfo_top_neighborhood[:10]
    df_expensive_neighborhoods.reset_index(inplace=True)
    df_expensive_neighborhoods_per_year = df_costs[df_costs["neighborhood"].
↪isin(df_expensive_neighborhoods["neighborhood"])]
    df_expensive_neighborhoods_per_year.reset_index(inplace=True)
    sunburst_plot = px.sunburst(
        df_expensive_neighborhoods_per_year,
        path=['year', 'neighborhood'],
        values='sale_price_sqr_foot',
        color='gross_rent',
        color_continuous_scale='blues',
        width=800,
        height=800,
```

```
        title='Cost Analysis of Most Expensive neighborhoods in San Francisco␣
↪per Year'
    )
    return sunburst_plot
```

2.2 Panel Dashboard

In this section, you will combine all of the plots into a single dashboard view using Panel. Be creative with your dashboard design!

[7]:
```
welcome_title = "### This dashboard represents a visual analysis of historical␣
↪prices of house units, sale price per square foot" \
                " and gross rent in San Francisco, California from 2010 to 2016.
↪  You can navigate through the tabs above to explore" \
                " more details about the evolution of the real estate market in␣
↪The Golden City across these years."
welcome = pn.Column(welcome_title, neighborhood_map())
yearly_market_analysis = pn.Row(housing_units_per_year, average_gross_rent,␣
↪average_sales_price)
neighborhood_analysis = pn.Column(average_price_by_neighborhood,␣
↪top_most_expensive_neighborhoods)
parallel_plots_analysis = pn.Column(parallel_coordinates, parallel_categories)
```

[8]:
```
# Create a Title for the Dashboard
dashboard_title = '# Real Estate Analysis of San Francisco from 2010 to 2016'

# Create a tab layout for the dashboard
sfo_market_dashboard = pn.Column(pn.pane.Markdown(dashboard_title, width=700),␣
↪pn.Tabs(
    ('Welcome', welcome),
    ('Yearly Market Analysis', yearly_market_analysis),
    ('Neighborhood Analysis', neighborhood_analysis),
    ('Parallel Plots Analysis', parallel_plots_analysis),
    ('Sunburst Plot Analysis', sunburst))
)

# Create the dashboard
sfo_market_dashboard.servable()
```

[8]:
```
Column
    [0] Markdown(str, width=700)
    [1] Tabs
        [0] Column
            [0] Markdown(str)
            [1] Plotly(Figure)
        [1] Row
            [0] Column
```

```
                [0] Column()
                [1] Row
                    [0] HoloViews(Bars, name='interactive01557')
            [1] Column
                [0] Column()
                [1] Row
                    [0] HoloViews(Curve, name='interactive01683')
            [2] Column
                [0] Column()
                [1] Row
                    [0] HoloViews(Curve, name='interactive01765')
        [2] Column
            [0] Column
                [0] Column()
                [1] Row
                    [0] Row(name='interactive01848')
                        [0] HoloViews(DynamicMap, name='interactive01848')
                        [1] Column
                            [0] WidgetBox
                                [0] Select(margin=(20, 20, 20, 20),
name='neighborhood', options=['Alamo Square', …], value='Alamo Square',
width=250)
                            [1] VSpacer()
            [1] Column
                [0] Column()
                [1] Row
                    [0] HoloViews(Bars, name='interactive01883')
        [3] Column
            [0] Column
                [0] Column()
                [1] Row
                    [0] Plotly(Figure, name='interactive02012')
            [1] Column
                [0] Column()
                [1] Row
                    [0] Plotly(Figure, name='interactive02018')
        [4] Column
            [0] Column()
            [1] Row
                [0] Plotly(Figure, name='interactive02027')
```

2.3 Serve the Panel Dashboard

[9]:
```
# Serve the# dashboard
sfo_market_dashboard.show()
```

```
Launching server at http://localhost:60725
```

[9]:
```
<bokeh.server.server.Server at 0x2b0565ce520>
```

3 Debugging

Note: Some of the Plotly express plots may not render in the notebook through the panel functions.

However, you can test each plot by uncommenting the following code

[69]:
```
# housing_units_per_year()
```

[68]:
```
# average_gross_rent()
```

[67]:
```
# average_sales_price()
```

[66]:
```
# average_price_by_neighborhood()
```

[65]:
```
# top_most_expensive_neighborhoods()
```

[64]:
```
# most_expensive_neighborhoods_rent_sales()
```

[63]:
```
# neighborhood_map().show()
```

[62]:
```
# parallel_categories()
```

[61]:
```
# parallel_coordinates()
```

[60]:
```
# sunburst()
```

[ ]:
```

```