5_Peer_Graded_Assignment_Questions

June 10, 2021

1 Assignment

- Section 1.1
- Section 1.2
- Section 1.3
- Section 1.4
- Section **??**
- Section 1.6
- Section 1.7
- Section 1.8

1.1 Story:

As a data analyst, you have been given a task to monitor and report US domestic airline flights performance. Goal is to analyze the performance of the reporting airline to improve fight reliability thereby improving customer relaibility.

Below are the key report items,

- Yearly airline performance report
- Yearly average flight delay statistics

NOTE: Year range is between 2005 and 2020.

1.2 Components of the report items

1. Yearly airline performance report

   For the chosen year provide,

   - Number of flights under different cancellation categories using bar chart.
   - Average flight time by reporting airline using line chart.
   - Percentage of diverted airport landings per reporting airline using pie chart.
   - Number of flights flying from each state using choropleth map.
   - Number of flights flying to each state from each reporting airline using treemap chart.

2. Yearly average flight delay statistics

   For the chosen year provide,

   - Monthly average carrier delay by reporting airline for the given year.
   - Monthly average weather delay by reporting airline for the given year.
   - Monthly average natioanl air system delay by reporting airline for the given year.

- Monthly average security delay by reporting airline for the given year.
- Monthly average late aircraft delay by reporting airline for the given year.

NOTE: You have worked created the same dashboard components in `Flight Delay Time Statistics Dashboard` section. We will be reusing the same.

1.3 Expected Layout

1.4 Requirements to create the dashboard

- Create dropdown using the reference here
- Create two HTML divisions that can accomodate two components (in one division) side by side. One is HTML heading and the other one is dropdown.
- Add graph components.
- Callback function to compute data, create graph and return to the layout.

1.5 What's new in this exercise compared to other labs?

- Make sure the layout is clean without any defualt graphs or graph layouts. We will do this by 3 changes:

  1. Add `app.config.suppress_callback_exceptions = True` right after `app = JupyterDash(__name__)`.

  2. Having empty html.Div and use the callback to Output the dcc.graph as the Children of that Div.

  3. Add a state variable in addition to callback decorator input and output parameter. This will allow us to pass extra values without firing the callbacks. Here, we need to pass two inputs `chart type` and `year`. Input is read only after user entering all the information.

- Use new html display style `flex` to arrange the dropdown menu with description.

- Update app run step to avoid getting error message before initiating callback.

NOTE: These steps are only for review.

1.6 Review

Search/Look for review to know how commands are used and computations are carried out. There are 7 review items.

- REVIEW1: Clear the layout and do not display exception till callback gets executed.
- REVIEW2: Dropdown creation.
- REVIEW3: Observe how we add an empty division and providing an id that will be updated during callback.
- REVIEW4: Holding output state till user enters all the form information. In this case, it will be chart type and year.
- REVIEW5: Number of flights flying from each state using choropleth
- REVIEW6: Return dcc.Graph component to the empty division
- REVIEW7: This covers chart type 2 and we have completed this exercise under Flight Delay Time Statistics Dashboard section

1.7 Hints to complete TODOs

1.7.1 TODO1

Reference link

- Provide title of the dash application title as `US Domestic Airline Flights Performance`.
- Make the heading center aligned, set color as `#503D36`, and font size as `24`. Sample: style={'textAlign': 'left', 'color': '#000000', 'font-size': 0}

1.7.2 TODO2

Reference link

Create a dropdown menu and add two chart options to it.

Parameters to be updated in `dcc.Dropdown`:

- Set `id` as `input-type`.
- Set `options` to list containing dictionaries with key as `label` and user provided value for labels in `value`.

  1st dictionary

  - label: Yearly Airline Performance Report
  - value: OPT1

  2nd dictionary

  - label: Yearly Airline Delay Report
  - value: OPT2

- Set placeholder to `Select a report type`.
- Set width as `80%`, padding as `3px`, font size as `20px`, text-align-last as `center` inside style parameter dictionary.

Skeleton:

```
dcc.Dropdown(id='....',
                 options=[
                         {'label': '....', 'value': '...'},
                         {'label': '....', 'value': '...'}
                         ],
                placeholder='....',
                style={....})
```

1.7.3 TODO3

Add a division with two empty divisions inside. For reference, observe how code under `REVIEW` has been structured.

Provide division ids as `plot4` and `plot5`. Display style as `flex`.

Skeleton

```
html.Div([
        html.Div([ ], id='....'),
        html.Div([ ], id='....')
        ], style={....})
```

1.7.4 TODO4

Our layout has 5 outputs so we need to create 5 output components. Review how input components are constructured to fill in for output component.

It is a list with 5 output parameters with component id and property. Here, the component property will be `children` as we have created empty division and passing in `dcc.Graph` after computation.

Component ids will be `plot1` , `plot2`, `plot2`, `plot4`, and `plot5`.

Skeleton

```
[Output(component_id='plot1', component_property='children'),
 Output(....),
 Output(....),
 Output(....),
 Output(....)]
```

1.7.5 TODO5

Deals with creating line plots using returned dataframes from the above step using `plotly.express`. Link for reference is here

Average flight time by reporting airline

- Set figure name as `line_fig`, data as `line_data`, x as `Month`, y as `AirTime`, color as `Reporting_Airline` and `title` as `Average monthly flight time (minutes) by airline`.

Skeleton

```
carrier_fig = px.line(avg_car, x='Month', y='CarrierDelay', color='Reporting_Airline', title='A
```

)

1.7.6 TODO6

Deals with creating treemap plot using returned dataframes from the above step using `plotly.express`. Link for reference is here

Number of flights flying to each state from each reporting airline

- Set figure name as `tree_fig`, data as `tree_data`, path as `['DestState', 'Reporting_Airline']`, values as `Flights`, colors as `Flights`, color_continuous_scale as `'RdBu'`, and title as `'Flight count by airline to destination state'`

Skeleton

```
tree_fig = px.treemap(data, path=['...', '...'],
                      values='...',
                      color='...',
                      color_continuous_scale='...',
                      title='...'
                )
```

1.8 Application

[ ]:

```
# Import required libraries
import pandas as pd
import dash
import dash_html_components as html
import dash_core_components as dcc
from dash.dependencies import Input, Output, State
from jupyter_dash import JupyterDash
import plotly.graph_objects as go
import plotly.express as px
from dash import no_update


# Create a dash application
app = JupyterDash(__name__)
JupyterDash.infer_jupyter_proxy_config()

# REVIEW1: Clear the layout and do not display exception till callback gets␣
 ↪executed
app.config.suppress_callback_exceptions = True

# Read the airline data into pandas dataframe
airline_data =  pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.
 ↪appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/
 ↪Data%20Files/airline_data.csv',
                            encoding = "ISO-8859-1",
                            dtype={'Div1Airport': str, 'Div1TailNum': str,
                                   'Div2Airport': str, 'Div2TailNum': str})


# List of years
year_list = [i for i in range(2005, 2021, 1)]

"""Compute graph data for creating yearly airline performance report

Function that takes airline data as input and create 5 dataframes based on the␣
 ↪grouping condition to be used for plottling charts and grphs.
```

```
Argument:

    df: Filtered dataframe

Returns:
   Dataframes to create graph.
"""
def compute_data_choice_1(df):
    # Cancellation Category Count
    bar_data = df.groupby(['Month','CancellationCode'])['Flights'].sum().
 ↪reset_index()
    # Average flight time by reporting airline
    line_data = df.groupby(['Month','Reporting_Airline'])['AirTime'].mean().
 ↪reset_index()
    # Diverted Airport Landings
    div_data = df[df['DivAirportLandings'] != 0.0]
    # Source state count
    map_data = df.groupby(['OriginState'])['Flights'].sum().reset_index()
    # Destination state count
    tree_data = df.groupby(['DestState', 'Reporting_Airline'])['Flights'].sum().
 ↪reset_index()
    return bar_data, line_data, div_data, map_data, tree_data


"""Compute graph data for creating yearly airline delay report

This function takes in airline data and selected year as an input and performs␣
 ↪computation for creating charts and plots.

Arguments:
    df: Input airline data.

Returns:
    Computed average dataframes for carrier delay, weather delay, NAS delay,␣
 ↪security delay, and late aircraft delay.
"""
def compute_data_choice_2(df):
    # Compute delay averages
    avg_car = df.groupby(['Month','Reporting_Airline'])['CarrierDelay'].mean().
 ↪reset_index()
    avg_weather = df.groupby(['Month','Reporting_Airline'])['WeatherDelay'].
 ↪mean().reset_index()
    avg_NAS = df.groupby(['Month','Reporting_Airline'])['NASDelay'].mean().
 ↪reset_index()
    avg_sec = df.groupby(['Month','Reporting_Airline'])['SecurityDelay'].mean().
 ↪reset_index()
```

```
    avg_late = df.groupby(['Month','Reporting_Airline'])['LateAircraftDelay'].
 ↪mean().reset_index()
    return avg_car, avg_weather, avg_NAS, avg_sec, avg_late


# Application layout
app.layout = html.Div(children=[
                                # TODO1: Add title to the dashboard


                                # REVIEW2: Dropdown creation
                                # Create an outer division
                                html.Div([
                                    # Add an division
                                    html.Div([
                                        # Create an division for adding␣
 ↪dropdown helper text for report type
                                        html.Div(
                                            [
                                            html.H2('Report Type:',␣
 ↪style={'margin-right': '2em'}),
                                            ]
                                        ),
                                        # TODO2: Add a dropdown

                                    # Place them next to each other using the␣
 ↪division style
                                    ], style={'display':'flex'}),

                                   # Add next division
                                   html.Div([
                                       # Create an division for adding dropdown␣
 ↪helper text for choosing year
                                        html.Div(
                                            [
                                            html.H2('Choose Year:',␣
 ↪style={'margin-right': '2em'})
                                            ]
                                        ),
                                        dcc.Dropdown(id='input-year',
                                                     # Update dropdown values␣
 ↪using list comphrehension
                                                     options=[{'label': i,␣
 ↪'value': i} for i in year_list],
                                                     placeholder="Select a␣
 ↪year",
                                                     style={'width':'80%',␣
 ↪'padding':'3px', 'font-size': '20px', 'text-align-last' : 'center'}),
                                            # Place them next to each other␣
 ↪using the division style
                                            ], style={'display': 'flex'}),
                                          ]),

                                # Add Computed graphs
                                # REVIEW3: Observe how we add an empty division␣
 ↪and providing an id that will be updated during callback
                                html.Div([ ], id='plot1'),

                                html.Div([
                                        html.Div([ ], id='plot2'),
                                        html.Div([ ], id='plot3')
                                ], style={'display': 'flex'}),

                                # TODO3: Add a division with two empty␣
 ↪divisions inside. See above disvision for example.

                                ])

# Callback function definition
# TODO4: Add 5 ouput components
@app.callback( [....],
               [Input(component_id='input-type', component_property='value'),
                Input(component_id='input-year', component_property='value')],
               # REVIEW4: Holding output state till user enters all the form␣
 ↪information. In this case, it will be chart type and year
               [State("plot1", 'children'), State("plot2", "children"),
                State("plot3", "children"), State("plot4", "children"),
                State("plot5", "children")
               ])
# Add computation to callback function and return graph
def get_graph(chart, year, children1, children2, c3, c4, c5):

        # Select data
        df =  airline_data[airline_data['Year']==int(year)]

        if chart == 'OPT1':
            # Compute required information for creating graph from the data
            bar_data, line_data, div_data, map_data, tree_data =␣
 ↪compute_data_choice_1(df)

            # Number of flights under different cancellation categories
            bar_fig = px.bar(bar_data, x='Month', y='Flights',␣
↪color='CancellationCode', title='Monthly Flight Cancellation')

            # TODO5: Average flight time by reporting airline


            # Percentage of diverted airport landings per reporting airline
            pie_fig = px.pie(div_data, values='Flights',␣
↪names='Reporting_Airline', title='% of flights by reporting airline')

            # REVIEW5: Number of flights flying from each state using choropleth
            map_fig = px.choropleth(map_data,  # Input data
                    locations='OriginState',
                    color='Flights',
                    hover_data=['OriginState', 'Flights'],
                    locationmode = 'USA-states', # Set to plot as US States
                    color_continuous_scale='GnBu',
                    range_color=[0, map_data['Flights'].max()])
            map_fig.update_layout(
                    title_text = 'Number of flights from origin state',
                    geo_scope='usa') # Plot only the USA instead of globe

            # TODO6: Number of flights flying to each state from each reporting␣
↪airline




            # REVIEW6: Return dcc.Graph component to the empty division
            return [dcc.Graph(figure=tree_fig),
                    dcc.Graph(figure=pie_fig),
                    dcc.Graph(figure=map_fig),
                    dcc.Graph(figure=bar_fig),
                    dcc.Graph(figure=line_fig)
                   ]
        else:
            # REVIEW7: This covers chart type 2 and we have completed this␣
↪exercise under Flight Delay Time Statistics Dashboard section
            # Compute required information for creating graph from the data
            avg_car, avg_weather, avg_NAS, avg_sec, avg_late =␣
↪compute_data_choice_2(df)

            # Create graph
            carrier_fig = px.line(avg_car, x='Month', y='CarrierDelay',␣
↪color='Reporting_Airline', title='Average carrrier delay time (minutes) by␣
↪airline')
```

```
            weather_fig = px.line(avg_weather, x='Month', y='WeatherDelay',␣
↪color='Reporting_Airline', title='Average weather delay time (minutes) by␣
↪airline')
            nas_fig = px.line(avg_NAS, x='Month', y='NASDelay',␣
↪color='Reporting_Airline', title='Average NAS delay time (minutes) by␣
↪airline')
            sec_fig = px.line(avg_sec, x='Month', y='SecurityDelay',␣
↪color='Reporting_Airline', title='Average security delay time (minutes) by␣
↪airline')
            late_fig = px.line(avg_late, x='Month', y='LateAircraftDelay',␣
↪color='Reporting_Airline', title='Average late aircraft delay time (minutes)␣
↪by airline')

            return[dcc.Graph(figure=carrier_fig),
                   dcc.Graph(figure=weather_fig),
                   dcc.Graph(figure=nas_fig),
                   dcc.Graph(figure=sec_fig),
                   dcc.Graph(figure=late_fig)]


# Run the app
if __name__ == '__main__':
    # REVIEW8: Adding dev_tools_ui=False, dev_tools_props_check=False can␣
↪prevent error appearing before calling callback function
    app.run_server(mode="inline", host="localhost", debug=False,␣
↪dev_tools_ui=False, dev_tools_props_check=False)
```

1.9 Summary

Congratulations for completing your dash and plotly assignment.

More information about the libraries can be found here

1.10 Author

Saishruthi Swaminathan

1.11 Changelog

Date	Version	Changed by	Change Description
12-18-2020	1.0	Nayef	Added dataset link and upload to Git

##

[ ]: