MS IN FINANCE CAPSTONE PROJECT

Interactive Web Data Dashboard Built in Python for Banks and Investment Banks

Tserendorj Tumenbayar

Contents

List of tables

List of figures

Introduction

In today's fast paced and highly competitive digital economy, the need to process data and to efficiently deliver it to the end-user is one of the most crucial components of information reporting. Interactive reporting allows end-users to be able to filter and select specific or desired information from the report and utilize them to support the decision-making process.

As this demand is ever increasing, software companies are creating numerous business intelligence products. We can divide software's for creating dashboard in three categories.

Table 1. Data visualization software

All-purpose	Business BI	Specialized Dataviz
Python	Tableau	Sankeymatic
R	Power PI	Gephi
Excel	Qlik	Flourish
others	IBM; Oracle; SAS; SAP, others	others

In addition, many financial companies' reporting units require their employees to able to understand and work on these products as a basic necessity. However, these products can be quite expensive and complex to both the employer and employee when the need to create advanced charts such as Sankey, Sunburst, etc. occur

Therefore, the main objective of this project is to create a free and easy to use web-based tool that can conduct industry analysis and peer comparison study based on publicly available financial data with select timeframes. When creating the dashboard for the abovementioned analysis tool, I have used the following multinational financial companies and their publicly available information:

- Citigroup Inc
- Bank of America Corporation
- Goldman Sachs Group
- JPMorgan Chase & Co
- Morgan Stanley

It is also important to note that pertaining to client confidentiality requirements, the client for this project shall be left anonymous.

Data Source and Methodology

For the above firms, I have used 10-K filings of each company from fiscal years 2010 to 2019. All relevant information for each company is obtained from EDGAR, the United States of America's Securities and Exchange Commission system that stands for Electronic Data Gathering, Analysis, and Retrieval. Additionally, each company's website was used with information in many different formats such as pdf, HTML, etc.

The most important advantage of this project is that it is based purely on the Python programming language. The main reason of selecting Python over other programming languages is the fact that

is relatively easy to learn and use for people with non-programming background. To extract information from 10-K filings, I have used the python modules such as:

- Re/Regular Expression: This is an expression that is used to search a string or set of strings. Basically, a defined search.
- Pandas: An analysis tool created by Wes McKinney that enables data manipulation from numerical sources of data, tables and time series.
- Beautiful Soup: Another tool used to pull data or parse HTML and XML documents

As I am looking at tabular data, these modules have powerful functions for processing and presenting tabular data.

In order to create interactive charts, I looked for modules on Python that are dedicated for this purpose. As a result, I found out that most commonly used modules for creating interactive dashboards are comprised of Bokeh and Plotly modules.

- Bokeh: This is an interactive visualization library for web browsers. The main advantages of using this library are its on point construction of graphics and its strength to maintain interactivity even when using large or live datasets.
- Plotly: Another commonly used Python library for web browsers that makes high quality, interactive graphs.

I chose the Plotly module to create a dashboard because it has a productive Python framework for building web applications which is Dash app. The primary modules are used to create an interactive chart, which means that users can select and filter series, values, and are able to zoom in or out and hover through the graphs.

Step by Step guide

This section shows step by step guide to create dashboard.

If you are new user of Python, following software's will help to work on python environment.

- Anaconda is the world's most popular Python distribution platform[1].
- JetBrains PyCharm is a Python IDE for data science and web development[2].

The next step is to install necessary packages which need to create dashboard. Once Anaconda installed, most of popular modules are installed with on your computer.

But Plotly and Dash modules won't be installed automatically, therefore we must install it manually.[3] The easy and efficient way to install the module is to use Anaconda command prompt. Choose it from windows and

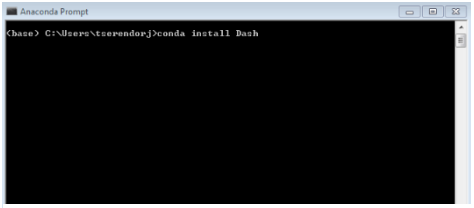

Figure 1. Anaconda command prompt

[1] https://www.anaconda.com/
[2] https://www.jetbrains.com/pycharm/
[3] If you are working on multiple project, highly recommend creating the virtual environment for each project. To do so each project have its own dependencies.

after it pops up, type following command in the window: conda install plotly; conda install dash.

Layout

After installing necessary packages, the first thing to do is to create web layout. Dash app consists of visual components which are dash_core_components, dash_html_components. Following example of code is used for control section of key ratios tabs of home page. First we are using *html.Div* to create a section which is located on the top left side of the page and width of this section will be 1/3 of total width of the page as we declare *className="pretty_container four columns"*.

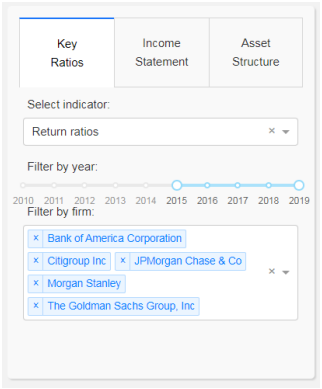

Figure 2. Control section of home page

Then next line of code says this section will have tabs, *dcc.Tabs*, which are Key Ratios, Income Statement, and Balance tab. Then it creates dropdown with the help of, *dcc.Dropdown*, menu for indicators. Options attribute must be list type and each element of the list must be dictionary type. After that, it creates slider, *dcc.RangeSlider*, from 2010 to 2019. Lastly, it creates another dropdown menu which is a little bit different from previous one. It can contain multiple value, *multi=True*. Also every id attribute value must be unique.

```
html.Div(
  [
    dcc.Tabs(id='tabs_example', value='tab_1', children=[
      dcc.Tab(label = 'Key Ratios',id='tab_1_', value='tab_1', children=[
  html.Div([
    html.P("Select indicator:", className="control_label"),
    dcc.Dropdown(
      id="indicator_status_1",
      options=[{"label": "Return ratios", "value": "ROE"},
            {"label": "Margin ratios & CAR", "value": "margin"},
      ],
      value="ROE",
      className="dcc_control",
    ),

    html.P(
      "Filter by year:",
      className="control_label",
    ),
    dcc.RangeSlider(
      id="year_slider",
      min=2010,
      max=2019,
      step=None,
      marks={2010: '2010', 2011:'2011', 2012:'2012', 2013:'2013', 2014:'2014', 2015:'2015',
          2016:'2016', 2017:'2017', 2018:'2018', 2019:'2019'},
      value=[2015, 2019],
      className="dcc_control",
    ),
    html.P("Filter by firm:", className="control_label"),
    dcc.Dropdown(
```

```
        id="firm_statuses",
        options=[{"label": "Bank of America Corporation", "value": "Bank of America Corporation"},
              {"label": "Citigroup Inc", "value": "Citigroup Inc"},
              {"label": "JPMorgan Chase & Co", "value": "JPMorgan Chase & Co"},
              {"label": "Morgan Stanley", "value": "Morgan Stanley"},
              {"label": "The Goldman Sachs Group, Inc", "value": "The Goldman Sachs Group, Inc"},
              ],
        multi=True,
        value=["Bank of America Corporation", "Citigroup Inc", "JPMorgan Chase & Co",
            "Morgan Stanley", "The Goldman Sachs Group, Inc"],
        className="dcc_control",
      ),
      ])
      ),
   ],
   className="pretty_container four columns",
   id="cross-filter-options",
),
```

For graph, below line of codes creates empty figure, *dcc.Graph*. We will connect this graph to database and control section using Callback.

```
html.Div(
   [dcc.Graph(id="second_graph1")],
   className="pretty_container five columns",
),
```

Callback

The CallBack makes the application interactive and dynamic. In simple words, it is like Excel programming. If you change input cell, dependent cells will get updated automatically. The Callback must begin special word, *@app.callback*, which contains single or multiple outputs and inputs. After that next line function must follow, otherwise it gives debug. It is also important to note that function arguments must follow input of the callbacks order. Below line of codes show basic simple callback. First it reads data from CSV file. The it is declaring the callback with only single output and input which means if we change slider value in the control section, the pie graph of region value will change. We must specify in the function how we would like to change it. In function we create two table from initial data using pandas' subsetting method, *dff = df_region[(df_region["Year"] == year_slider[1])].* After that, it creates empty figure for 2 pie charts. Then it adds value from two table which we created and makes fancy.

```
df_region = pd.read_csv(DATA_PATH.joinpath("JPM_revenue_region.csv"))

@app.callback(
   Output("second_graph1", "figure"),
   [
      Input("year_slider", "value"),
   ],
)
def make_second_graph(year_slider):
   dff = df_region[(df_region["Year"] == year_slider[1])]
   dfk = df_region[(df_region["Year"] == year_slider[0])]
```

```
    fig = make_subplots(rows=1, cols=2, specs=[[{'type': 'domain'}, {'type': 'domain'}]])
    fig.add_trace(go.Pie(labels=dfk["Region"], values=dfk["Value"], sort=False, name=year_slider[0]), 1, 1)
    fig.add_trace(go.Pie(labels=dff["Region"], values=dff["Value"], sort=False, name=year_slider[1]), 1, 2)
    fig.update_traces(hole=.4, hoverinfo="label+percent+name")

    fig.update_layout(
        title_text="Revenue by region: {} vs {}".format(year_slider[0], year_slider[1]),
        # Add annotations in the center of the donut pies.
        annotations=[dict(text="Year " + str(year_slider[0]), x=0.14, y=0.5, font_size=9, showarrow=False),
                     dict(text="Year " + str(year_slider[1]), x=0.86, y=0.5, font_size=9, showarrow=False)])
    fig.update_layout(
        paper_bgcolor='rgba(0,0,0,0)',
        plot_bgcolor='rgba(0,0,0,0)'
    )
    fig.update_layout(legend_orientation="h")
    return fig
```

Not only control section values can change graph's value but also hovering graph can change another graph's value. When users change user-interaction on the graph, it updates four attributes such as hoverData, clickData, selectedData, relayoutData. These four attributes return JSON, Javascript Object Notation, type value which is similar like nested dictionary type in Python. In this dashboard, hoverData has been used in the Morgan Stanley's product graph.

```
@app.callback(Output("product_graph", "figure"),
              [
                 Input("segment_graph", "hoverData"),
                 Input("year_slider", "value"),
              ],
)
def make_individual_figure(hoverData, year_slider):
    if hoverData is None:
        hoverData = {
            "points": [
              {"curveNumber": 1, "pointNumber": 0}
            ]
        }
    if hoverData["points"][0]["curveNumber"] == 2:
        a = ["IM"]
    elif hoverData["points"][0]["curveNumber"] == 1:
        a = ["WM"]
    else:
        a = ["IS"]


fig = go.Figure(data=[
        go.Bar(name = "Investment banking", x=d_IB["o_year"], y=d_IB["Value"], ),
        .........
        go.Bar(name = "Net interest", x=d_NI["o_year"], y=d_NI["Value"], ),

    ], )

    fig.update_xaxes(type='category')
    return fig
```

Following figure shows all callbacks used in this Dashboard

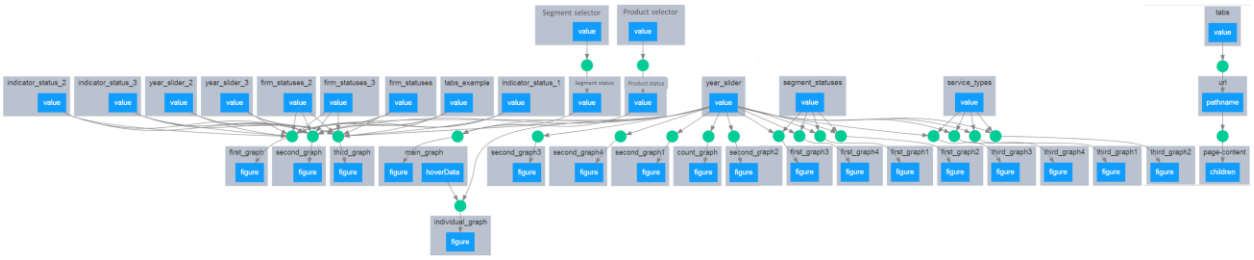

Figure 3. Structure of Callbacks

Multi-page apps

The way of rendering web application in Dash is single-page app. Out of the numerous ways of structuring multi-page app following structure is the simplest and user-friendly.

-app.py

-index.py

-apps

| - - __init__.py

| - - Home.py

| - - MorganStanley.py

| - - JPMorgan.py

| - - GoldmanSachs.py

| - - BofA.py

| - - Citi.py

The following code shows multipage app structure and navigation. First, it imports app which contains Dash framework then it imports other pages. Dcc.Location will give us opportunity to navigate URL. Basically, following callbacks works like chain. When users select, it changes URL value then it calls corresponding layout of pages. It is also worth to mention that if you work on multiple pages simultaneously, indicating port value is good practice, app.run_server(debug=True, port = 8051).

```
from app import app
from apps import Home, MorganStenley, JPMorgan, GoldmanSachs, BofA, Citi

server = app.server

app.layout = html.Div([
   dcc.Tabs(id='tabs', value='tab_1', children=[
      dcc.Tab(label='Home', value='tab_1'),
      dcc.Tab(label='Morgan Stanley', value='tab_2'),
      dcc.Tab(label='JPMorgan Chase & Co', value='tab_3'),
      dcc.Tab(label='The Goldman Sachs Group', value='tab_4'),
      dcc.Tab(label='The Bank of America Corporation', value='tab_5'),
      dcc.Tab(label='Citigroup Inc', value='tab_6'),
```

```
    ]),
    dcc.Location(id='url', refresh=False),
    html.Div(id='page-content')
])

@app.callback(Output('url', 'pathname'),
              [Input('tabs', 'value')])
def tabss(tabs):
    if tabs == 'tab_1':
        return '/home'
    elif tabs == 'tab_2':
        return '/MS'
    elif tabs == 'tab_3':
        return '/JPM'
    elif tabs == 'tab_4':
        return '/GS'
    elif tabs == 'tab_5':
        return '/BAC'
    elif tabs == 'tab_6':
        return '/C'
    else:
        return '404'

@app.callback(Output('page-content', 'children'),
              [Input('url', 'pathname')])
def kjhkjh(pathname):
    if pathname == '/home':
        return Home.layout
    elif pathname == '/MS':
        return MorganStenley.layout
    elif pathname == '/JPM':
        return JPMorgan.layout
    elif pathname == '/GS':
        return GoldmanSachs.layout
    elif pathname == '/BAC':
        return BofA.layout
    elif pathname == '/C':
        return Citi.layout
    else:
        return '404'

if __name__ == '__main__':
    app.run_server(debug=True)
```

Result and Usage

This dashboard consists of 6 pages. Each page has a control board which is located in the top left side. In order to maintain user friendly interface, each page contains only 3 charts.

The home page shows a comparison of the selected firm's information. Users can select from three different tabs which are key ratios, income, and balance. Each tab contains 2 type indicators *(return ratios, margin ratios and CAR)* which allows user to select their desired information.

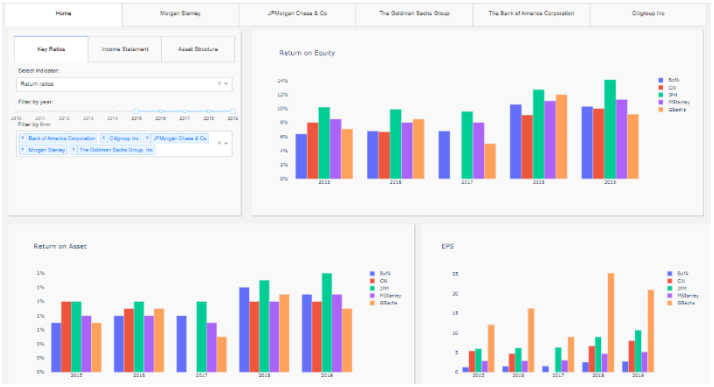

Figure 4. Home page of Dashboard

In the home page, users are able to select key ratios such as Return of Assets (ROA) and Return on Equity (ROE) for deeper analysis purposes. For example, if the user requires a 10 year comparison of Morgan Stanley and Goldman Sachs, the user can filter the duration by selecting 2010, 2019 and only selecting the above companies in the Filter by Firm section.

Below table shows the indicators which are in the home page. Additionally, users can filter by year and firms.

Table 2. Home page graphs

Tabs	**Key Ratios**		**Income Statement**		**Balance**	
Indicators	Return Ratios	Margin Ratios and CAR	Income	Expense	Asset	Liability
Chart 1	ROE	SG&A margin	Total revenue	Total non-interest expense	Total Asset	Total liability
Chart 2	ROA	Net income margin	Net interest income	Interest tax expense	Trading Securities	Short-term borrowings
Chart 3	EPS	TIER I ratio	Net income	Interest expense	Total Loans	Market Capitalization

The remaining 5 pages contain each firm's revenue structure. Users can see and analyze the revenue sources by region, segments, and products each in different graphs presented in the three boxes per page. Also, it allows filtering by year, segments, and products. Below tables show business segments of each firm.

Table 3. Business Segments

Morgan Stanley	JPMorgan Chase & Co	The Goldman Sachs Group	The Bank of America Corporation	Citigroup Inc
Institional Securities	Consumer & Community Banking	Investment Banking	Consumer Banking - Deposits	Global Consumer Banking

Wealth Management	Corporate & Investment Bank	Global Markets	Consumer Banking - Consumer Lending	Institional Clients Group
Investment Management	Commercial Banking	Asset Management	Global Wealth & Investment Management	Corporate/Other
	Asset & Wealth Management	Consumer & Wealth Management	Global Banking	Citi Holdings
	Corporate Segment	Institutional Client Services	Global Markets	
	Other	Investing & Lending	Other	
		Investment Management		

For the lower left box, the user can hover the mouse over each bar to view more detailed information regarding net revenue for the three business segments. In addition, the lower right box can display a further breakdown of each business segment by its product type.

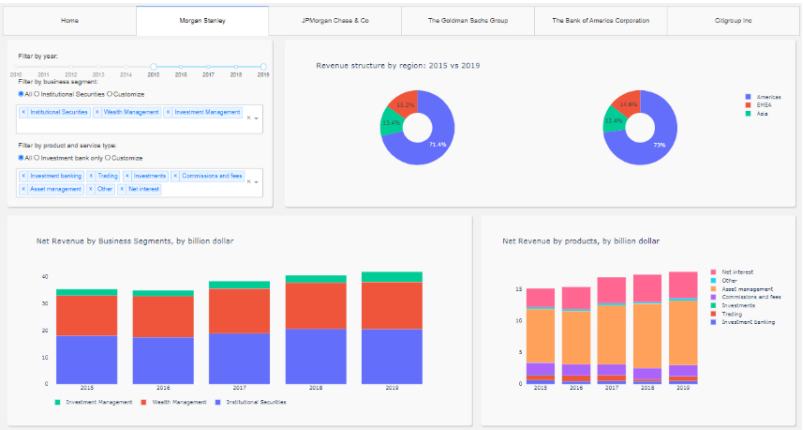

Figure 5. Page of Morgan Stanley

The dashboard is available for use and analysis at www.tserendorj.pythonanywhere.com, which is a free public server.

Conclusion

With rapidly developing digital economy, an essential part of information reporting is not only to process data but also to deliver it to end-users in an engaging manner. Correspondingly, a broad range of business intelligence products are accessible, however, the products come a high price per license and require a certain level of expertise. This line of thought gave me an incentive to create cost efficient, interactive reporting tool where end-users can filter on specific criteria and access the information, they want with precision they need.

Therefore, primary purpose of this project is to provide a free dashboard analysis tool that is based on Python programming language. Users can also assess with other business intelligence tools for comparative reasons. Within the scope of completing this project, I was able to sharpen my programming skills and also develop business intelligence better. In addition, I have better understanding and knowledge of multinational financial firms.

References

https://www.sec.gov/edgar.shtml

https://www.morganstanley.com/about-us-ir

https://www.jpmorganchase.com/corporate/investor-relations/investor-relations.htm

https://www.goldmansachs.com/investor-relations/

http://investor.bankofamerica.com/investor-relations

https://www.citigroup.com/citi/investor/sec.htm

https://plotly.com/python/

https://dash.plotly.com/