1. Introduction

Plotly Dash gives us the capablility to design a web-based dashboard that allows user input to decide what is shown on the screen. For example, We can have multiple plots that interact amongst themselves depending where the user is hovering the mouse. It can als oallow for other forms of input, such as dropdowns, radio buttons, text entry, and much more.

In [1]:
```
# Prepare datasets

import pandas as pd

df = pd.read_csv('customer_dataset.csv')
df.head()
```

Out[1]:

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen	pca_x	pca_y	umap_x	umap_y	ae_x	ae_y	vae_x
0	2	3	12669	9656	7561	214	2674	1338	0.193291	-0.305100	7.084310	6.933166	3.548878	3.811006	0.828640
1	2	3	7057	9810	9568	1762	3293	1776	0.434420	-0.328413	6.252880	7.050780	3.579156	2.955884	0.838629
2	2	3	6353	8808	7684	2405	3516	7844	0.811143	0.815096	8.588828	6.877347	1.341199	2.187068	0.841106
3	1	3	13265	1196	4221	6404	507	1788	-0.778648	0.652754	13.654358	7.857928	6.349530	8.099434	0.814431
4	2	3	22615	5410	7198	3915	1777	5185	0.166287	1.271434	9.122227	5.977852	1.150562	3.304798	0.853156

Useful Source

How to dash

Dash Apps은 크게 'laout'과 'application'으로 구성됩니다. Dash는 application의 가시화 할 요소들을 Python class로 제공한다.

Layout에서 주목할점

1. layout은 components로 구성되어있다. components 는 html.Div, doc.Graph와 같은 것들이 있다.
2. dash_html_components 라이브러리에는 모든 HTLM tag가 존재한다.

2. Construction

2.1 Importing library

In [4]:
```
import dash
import dash_core_components as dcc
import dash_bootstrap_components as dbc
import dash_html_components as html
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
import numpy as np
from jupyter_dash import JupyterDash
```

2.2 Basic Style Sheet

In [5]:
```
external_stylesheets = [dbc.themes.DARKLY]
app = JupyterDash(__name__, title = 'Interactive Model Dashboard', external_stylesheets = [external_stylesheets])
```

main 함수에서 실행되는 app 인스턴스는 dash.Dash() 객체입니다. app name을 변경하고 싶으면 pytho 파일의 이름을 바꾸는 것이 아닌 이부분을 바꿔야 합니다.

2.3 Background Dataset

In [6]:
```
df = pd.read_csv('customer_dataset.csv')
features = ['Fresh', 'Milk', 'Grocery', 'Frozen', 'Detergents_Paper', 'Delicatessen']
modles = ['PCA', 'UMAP', 'AE', 'VAE']
df_average = df[features].mean()
max_val = df[features].max().max()
```

2.4 Basic frame

- html.Div : 기본적인 사각 프레임을 구성한다.
- html.Label(' ') : html.Div에 들어갈 Text를 입력한다.
- dcc.Dropdown() : Selection box에 들어갈 구성요소를 생성한다.
  - id = ' ' : code가 인식할 Div id를 생성한다.
  - options = [label' : , 'value' :] : label은 User가 인식할내용 value는 코드가 인식할 내용이다.
  - value = ' ' : Default check
- dcc.Graph() : plot이 들어갈 구성요소를 생성한다.
  - id = ' ' : code가 인식할 Div id를 생성한다.

In [7]:
```
app.layout = html.Div([  # whole block

    html.Div([  # first block

        html.Div([     # first - left block

            html.Div([html.Label('Model selection')], style = {'font-size' : '18px'}),      # selection box

            dcc.Dropdown(                                                                   # selection value

                id = 'crossfilter-model',
                options = [
                    {'label' : 'Principal Component Analysis', 'value' : 'PCA'},
                    {'label' : 'Uniform Manifold Approximation and Projection', 'value' : 'UMAP'},
                    {'label' : 'Autoencoder', 'value' : 'AE'},
                    {'label' : 'Variational Autoencoder', 'value' : 'VAE'}
                ],
                value = 'PCA',
                clearable = False
            )
        ], style = {'width' : '49%', 'display' : 'inline-block'}),


        html.Div([       # first - right block

            html.Div([html.Label('Feature selection')], style = {'font-size' : '18px', 'width' : '40%', 'display' :
'inline-block'}),   # selection box

            html.Div([

                dcc.RadioItems(
                    id = 'gradient-scheme',
                    options = [
                        {'label' : 'Orange to Red', 'value' : 'OrRd'},
                        {'label' : 'Viridis', 'value' : 'Viridis'},
                        {'label' : 'Plasma', 'value' : 'Plasma'},
                    ],
                    value = 'Plasma',
                    labelStyle = {'float' : 'right', 'display' : 'inline-block', 'margins-right' : 10}
                )
            ], style = {'width' : '49%', 'display' : 'inline-block', 'float' : 'right'}),

            dcc.Dropdown(
                id = 'crossfilter-feature',
                options = [{'label' : i, 'value' :i} for i in features + ['None', 'Region', 'Channel', 'Total_Spend'
]],

                value = 'None',
                clearable = False)
        ], style = {'width' : '49%', 'float' : 'right', 'display' : 'inline-block'}

        )
    ], style = {'backgroundCOlor' : 'rgb(17, 17, 17)', 'padding' : '10px 5px'}),

    # ------------------------------------------------------------------------------------------------------------
-------------------------------------- #
    html.Div([     # second block

        dcc.Graph(
            id = 'scatter-plot',
            hoverData = {'points' : [{'customdata' : 0}]}
        )
    ], style = {'width' : '100%', 'height' : '90%', 'display' : 'inline-block', 'padding' : '0 20'}),

    # ------------------------------------------------------------------------------------------------------------
-------------------------------------- #
    html.Div([     # third block

        dcc.Graph(
            id = 'point-plot'
        )
    ], style = {'display' : 'inline-block', 'width' : '100%'}),

], style = {'backgroundColor' : 'rgb(17, 17, 17)'})
```

2.5 Plot call back

Callback함수는 어떤 동작이 일어날 때마다 호출이 되는 함수이다. Output은 넣을 Div의 id와 동일해야한다. Input값은 callback함수 밑의 인자에 순서대로 입력된다.

In [8]:
```
@app.callback(
    dash.dependencies.Output('scatter-plot', 'figure'),
    [
        dash.dependencies.Input('crossfilter-feature', 'value'),
        dash.dependencies.Input('crossfilter-model', 'value'),
        dash.dependencies.Input('gradient-scheme', 'value')
    ]
)

def update_graph(feature, model, gradient) :

    if feature == 'None' :
        cols = None
        sizes = None
        hover_names = [f'Customer {ix}' for ix in df.index]

    elif features in ['Region', 'Channel'] :
        cols = df[feature].astpye(str)
        sizes = None
        hover_names = [f'Customer {ix}' for ix in df.index]

    else :
        cols = df[feature]
        sizes = [np.max([max_val, val]) for val in df[feature].values]
        hover_names = []
        for ix, val in zip(df.index.values, df[feature].values) :
            hover_names.append(f'Customer {ix}<br>{feature} value of {val}')

    fig = px.scatter(df,
                     x = df[f'{model.lower()}_x'],
                     y = df[f'{model.lower()}_y'],
                     opacity = 0.8,
                     template = 'plotly_dark',
                     color_continuous_scale = gradient,
                     hover_name = hover_names, color = cols, size = sizes)

    fig.update_traces(customdata = df.index)

    fig.update_layout(
        height = 650,
        hovermode = 'closest',
        template = 'plotly_dark'
    )

    fig.update_xaxes(showticklabels = False)
    fig.update_yaxes(showticklabels = False)

    return fig

def create_point_plot(df, title) :
    fig = go.Figure(
            data = [
                go.Bar(name = 'Average', x = features, y = df_average.values, marker_color = '#c178f6'),
                go.Bar(name = title, x = features, y = df.values, marker_color = '#89efbd')
            ]
    )
    fig.update_layout(
        barmode = 'group',
        height = 220,
        margin = {'l' : 20, 'b' : 30, 'r' : 10, 't' : 10},
        template = 'plotly_dark'
    )
    fig.update_xaxes(showgrid = False)
    fig.update_yaxes(type = 'log', range = [0, 5])
    return fig

@app.callback(
    dash.dependencies.Output('point-plot', 'figure'),
    [
        dash.dependencies.Input('scatter-plot', 'hoverData')
    ]
)

def update_point_plot(hoverData) :
    index = hoverData['points'][0]['customdata']
    title = f'Customer {index}'
    return create_point_plot(df[features].iloc[index], title)
```

2.6 Run dash

In [10]:
```
app.run_server(mode='external', port = 8052)
```

```
Dash app running on http://127.0.0.1:8052/
```