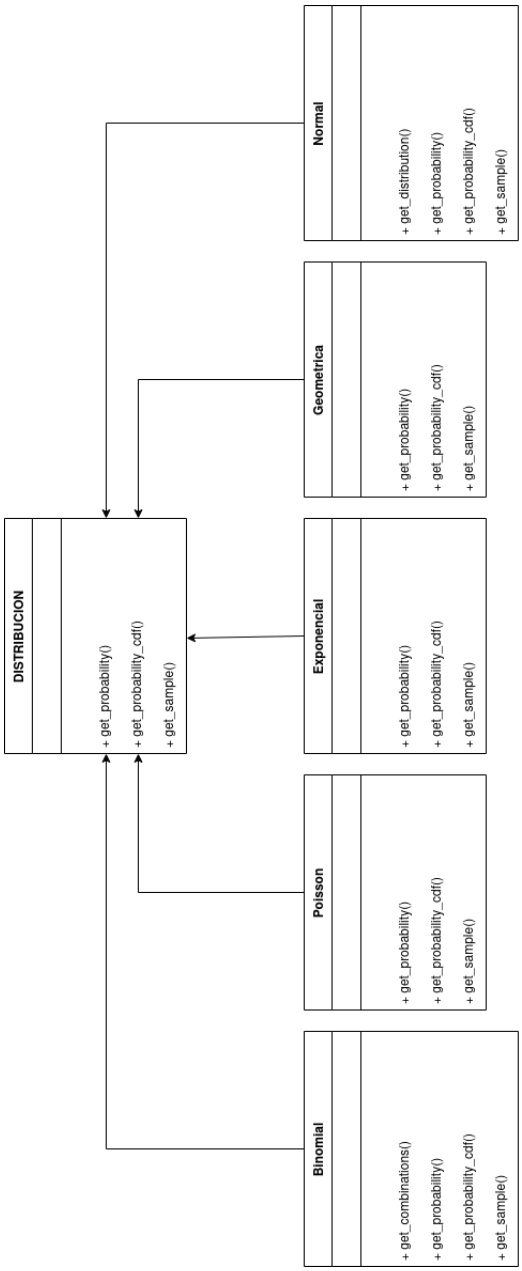
CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL IPN
UNIDAD TAMAULIPAS

Assignment #3

Student: *Luis Alberto Ballado Aradias, David Guadalupe Cervantes Martinez*
Course: *Tecnologías Computacionales (Sep - Dec 2022)*
Professor: *Dr. Edwyn Aldana Bobadilla*
December 20, 2022

Diagrama de Clases



Vista Proyecto Dash

El proyecto consta de tres archivos

- *server.py* - Archivo donde se contruyen los elementos visuales con ayuda del framework Dash en python
- *distribuciones.py* - Archivo donde se implementa el patron factory para cinco distribuciones (Binomial, Poisson, Geometrica, Exponencial, Normal
- *textos_distribuciones.py* - Archivo donde existe una función que regresa textos informativos por cada distribución

server.py

Al hacer uso de una extención de dash para tener acceso a componentes de tipo bootstrap se configuró lo siguiente:

```
1  import dash_bootstrap_components as dbc # importamos el paquete
2
3  external_stylesheets = [dbc.themes.BOOTSTRAP, dbc.icons.BOOTSTRAP] #
    configuramos que tome los estilos de bootstrap asi como los iconos
4
5  app = Dash(
6    __name__,
7    meta_tags=[{"name": "viewport", "content": "width=device-width,
    initial-scale=1"}],
8    external_stylesheets=external_stylesheets,
9    title="Distribuciones"
10 )
11
12 app._favicon = ("assets/favicon.ico") # colocamos un favicon
13
14 d = DistribucionFactory() # Se inicializa el objeto abstracto de las
    distribuciones
```

Código 1: Configuración Dash

Componentes DASH - Dropdown (selección tipo distribución)

```
1  # Bloque del dropdown
2  type_distribution = html.Div(
3      [
4          dbc.Label("Distribucion -"),
5          html.I(className="bi bi-info-circle-fill me-2", id="
    distribucion_tooltip"),
6          dcc.Dropdown(
7              id="tipo_distribucion",
8              options=[
9                  {"label": col, "value": col} for col in distribuciones
10             ],
11             value="Binomial",
12         ),
13         dbc.Tooltip("Selecciona un tipo de distribucion", target="
    distribucion_tooltip")
14     ]
15 )
```

Componentes DASH - Switch (seleccionar acumulada o no)

```
# Bloque del switch
switches = html.Div(
    [
        dbc.Label("Acumulada -"),
        html.I(className="bi bi-info-circle-fill me-2", id="acumulada_tooltip"),
        dbc.Switch(
            id="acumulada_switch",
            value=False,
        ),
        dbc.Tooltip("Calcular la version de probabilidad acumulada", target="acumulada_tooltip")
    ]
)
```

Componentes DASH - Inputs para distribucion binomial

```
# Bloque del switch
switches = html.Div(
    [
        dbc.Label("Acumulada -"),
        html.I(className="bi bi-info-circle-fill me-2", id="acumulada_tooltip"),
        dbc.Switch(
            id="acumulada_switch",
            value=False,
        ),
        dbc.Tooltip("Calcular la version de probabilidad acumulada", target="acumulada_tooltip")
    ]
)
```

Componentes DASH - Inputs para distribucion binomial

```
# Bloque de los parametros
parameters_binomial = html.Div(
    [
        dbc.Label("Parametros"),
        dbc.Input(id="binom_valor_n", placeholder="numero de pruebas", type="number"),
        html.Br(),
        dbc.Input(id="binom_valor_p", placeholder="probabilidad de exitos [0-1]", type="number"),
        html.Br(),
        html.Hr(),
        dbc.Label("Generar Valores para Graficar"),
        dbc.Input(id="binom_valor_x", placeholder="cardinalidad", type="number"),
    ], style= {'display': 'none'}, id='parameters_binomial'
)
```

Componentes DASH - Inputs para distribucion poisson

```
# Bloque de parametros poisson
parameters_poisson = html.Div(
    [
        dbc.Label("Parametros"),
        dbc.Input(id="pois_valor_mu", placeholder="numero de ocurrencias esperadas > 0", type="number"),
        html.Br(),
        html.Hr(),
        dbc.Label("Generar Valores para Graficar"),
        dbc.Input(id="pois_valor_x", placeholder="cardinalidad", type="number")
    ], style= {'display': 'none'}, id='parameters_poisson'
)
```

Componentes DASH - Inputs para distribucion geometrica

```
#Bloque de parametros geometrica
parameters_geometrica = html.Div(
    [
        dbc.Label("Parametros"),
        dbc.Input(id="geom_valor_p", placeholder="probabilidad de exito [0-1]", type="number"),
        html.Br(),
        html.Hr(),
        dbc.Label("Generar Valores para Graficar"),
        dbc.Input(id="geom_valor_x", placeholder="cardinalidad", type="number")
    ], style= {'display': 'none'}, id='parameters_geometrica'
)
```

Componentes DASH - Inputs para distribucion exponencial

```

# Bloque de parametros exponencial
parameters_exponencial = html.Div(
    [
        dbc.Label("Parametros"),
        dbc.Input(id="exp_valor_alpha", placeholder="tasa de ocurrencia del evento > 0", type="number"),
        html.Br(),
        html.Hr(),
        dbc.Label("Generar Valores para Graficar"),
        dbc.Input(id="exp_valor_x", placeholder="cardinalidad", type="number")
    ], style= {'display': 'none'}, id='parameters_exponencial'
)
```

Componentes DASH - Inputs para distribucion normal

```
# Bloque de parametros normal
parameters_normal = html.Div(
    [
        dbc.Label("Parametros"),
        dbc.Input(id="norm_valor_mu", placeholder="media de distribucion
    ", type="number"),
        html.Br(),
        dbc.Input(id="norm_valor_sigma", placeholder="desviacion
    estandar de la distribucion > 0", type="n\
umber"),
        html.Br(),
        html.Hr(),
        dbc.Label("Generar Valores para Graficar"),
        dbc.Input(id="norm_valor_x", placeholder="cardinalidad", type="
    number")
    ], style= {'display': 'none'}, id='parameters_normal'
)
```

Componentes DASH - Botones Aceptar / Cancelar

```
# Bloque de botones
actions_buttons = html.Div(
    [
        dbc.Button("CANCELAR", id="cancel_btn", color="danger",
    className="me-1", n_clicks=0),
        dbc.Button("ACEPTAR", id="aceptar_btn", color="success",
    className="me-1", n_clicks=0),
        html.Span(id="example-output", style={"verticalAlign": "middle",
    "display":"none"}),
    ],
    className="d-grid gap-2 d-md-flex justify-content-md-center",
)
```

Componentes DASH - Tarjeta con componentes

Al tener los componentes por partes, dentro de la tarjeta se agrupan de la siguiente manera

```
# Tarjeta con todos los elementos
controls = dbc.Card(
    [
        type_distribution,
        html.Br(),
        switches,
        html.Br(),
        parameters_binomial,
        parameters_poisson,
        parameters_geometrica,
        parameters_exponencial,
        parameters_normal,
        html.Br(),
        actions_buttons
    ],
    body=True,
)
```

Componentes DASH - Grafica

Por practicidad se inicializa un componente tipo gráfica ya que es necesario que este inicializado para poder mutarlo durante el proceso de cálculos. Así también se incluye un bloque de texto informativo para cada distribución.

```
# Grafica
generate_graph = dcc.Graph(
    id='distribution-graph',
    mathjax=True
)
```

```
# Bloque de texto
_latex_ = dcc.Markdown(
    children = ''' ''',
    mathjax=True,id='latex_text'
)
```

Contenedor Principal

Al tener los elementos por partes, dentro del layout en dash se incluyen las partes para formar la vista a mostrarse. Con uso de los estilos de bootstrap en el acomodo por renglones y columnas

```
# Contenedor principal
app.layout = dbc.Container(
    [
        html.H1("Distribuciones"),
        html.Hr(),
        dbc.Row(
            [
                dbc.Col(controls, md=4),
                dbc.Col(
                    generate_graph
                    , md=8),
            ],
            align="center",
  ),
        html.Hr(),
        html.Br(),
        _latex_
    ],
    fluid=True,
)
```

Dash - Callbacks

Parte de la funcionalidad descansa en el uso de los callbacks y poder mutar la vista con la información generada

Callback Dropdown - Este callback controlará los elementos visibles para ser elegidos por el usuario, de manera que no todas las distribuciones tienen las mismas entradas. Se define el callback como salidas los componentes de cada distribucion y como entrada el valor del dropdown

```
#Callback del dropdown
#oculta los inputs de las distribuciones
#que no fueron elegidos
@app.callback(
    Output('parameters_binomial', 'style'),
    Output('parameters_poisson', 'style'),
    Output('parameters_geometrica', 'style'),
    Output('parameters_exponencial', 'style'),
    Output('parameters_normal', 'style'),
    [Input(component_id='tipo_distribucion', component_property='value')
    ])
def show_hide_element(value):
    if value == 'Binomial':
        return {'display':'block'},{'display':'none'},{'display':'none'
    },{'display':'none'},{'display':'no\
ne'}
    elif value == 'Poisson':
        return {'display':'none'},{'display':'block'},{'display':'none'
    },{'display':'none'},{'display':'no\
ne'}
    elif value == 'Geometrica':
        return {'display':'none'},{'display':'none'},{'display':'block'
    },{'display':'none'},{'display':'no\
ne'}
    elif value == 'Exponencial':
        return {'display':'none'},{'display':'none'},{'display':'none'
    },{'display':'block'},{'display':'no\
ne'}
    elif value == 'Normal':
        return {'display':'none'},{'display':'none'},{'display':'none'
    },{'display':'none'},{'display':'blo\
ck'}
    else:
        return {'display':'none'},{'display':'none'},{'display':'none'
    },{'display':'none'},{'display':'non\
e'}
```

Código 2: Callback dropdown

Callback de Cancelar

Dentro de las primeras implementaciones de los callbacks fue de un boton cancelar que limpiara los inputs cuya entrada es el evento del click, y la salida son todos los inputs de todas las distribuciones

```
# Callback del boton CANCELAR
# Limpia todos los inputs
@app.callback(
    Output('acumulada_switch','value'),
    Output('binom_valor_n','value'),
    Output('binom_valor_p','value'),
    Output('binom_valor_x','value'),
    Output('pois_valor_mu','value'),
    Output('pois_valor_x','value'),
    Output('geom_valor_p','value'),
    Output('geom_valor_x','value'),
    Output('exp_valor_alpha','value'),
    Output('exp_valor_x','value'),
    Output('norm_valor_mu','value'),
    Output('norm_valor_sigma','value'),
    Output('norm_valor_x','value'),
    [Input('cancel_btn', 'n_clicks')]
)
def on_cancel_click(n):
    if n is None:
        return "Not clicked."
    else:
        return False,'','','','','','','','','','','',''
```

Código 3: Callback Boton Cancelar

El callback del boton aceptar es el mas cargado ya que contiene todos los calculos de las cinco distribuciones implementadas

Se cuenta como entrada la accion del boton aceptar, y como salida la generacion del grafico y el texto explicativo. Se observan los estados de los diferentes inputs de las distribuciones. Al ser pulsado el boton ACEPTAR y se tenga elegida una distribucion entrara dentro de la segunda parte del if-else

```
# Callback del buton ACEPTAR
# El output es la grafica
@app.callback(
    Output('distribution-graph', 'figure'),
    Output('latex_text', 'children'),
    Input('aceptar_btn', 'n_clicks'),
    State('tipo_distribucion','value'),
    State('acumulada_switch', 'value'),
    State('binom_valor_n','value'),
    State('binom_valor_p','value'),
    State('binom_valor_x','value'),
    State('pois_valor_mu','value'),
    State('pois_valor_x','value'),
    State('geom_valor_p','value'),
    State('geom_valor_x','value'),
    State('exp_valor_alpha','value'),
    State('exp_valor_x','value'),
    State('norm_valor_mu','value'),
    State('norm_valor_sigma','value'),
    State('norm_valor_x','value'),
)
def on_accept_click(n,tipo_distribucion,acumulada_switch,
                    binom_valor_n,binom_valor_p,binom_valor_x,
                    pois_valor_mu,pois_valor_x,
                    geom_valor_p,geom_valor_x,
                    exp_valor_alpha,exp_valor_x,
                    norm_valor_mu,norm_valor_sigma,norm_valor_x
                    ):

    if n <= 0:
        # Crear un grafico vacio
        text = ''''''
        return go.Figure(go.Scatter(x=[0], y=[0], fill="toself")),text
```

Código 4: Callback Boton Aceptar

El cálculo de la distribución binomial descansa en la clase abstracta y el uso del DataFrame de pandas

```
    else:

        # en acumulada se deben de sumar la prob actual mas la anterior

        if tipo_distribucion == 'Binomial':

            datos = {}
            datos['n'] = binom_valor_n
            datos['p'] = binom_valor_p
            datos['x'] = binom_valor_x
            datos['acumulada'] = acumulada_switch

            binomial = d.getDistribution("Binomial",datos)

            # Crear DataFrame
            if not acumulada_switch:
                df = pd.DataFrame(binomial.get_sample(binom_valor_x),
columns=['n_gen'])
                df['pdf'] = df['n_gen'].apply(lambda x: binomial.
get_probability(x))
                x_axis = "X_AXIS"
                y_axis = "Y_AXIS"
            else:
                df = pd.DataFrame(binomial.get_sample(binom_valor_x),
columns=['n_gen'])
                df['pdf'] = df['n_gen'].apply(lambda x: binomial.
get_probability_cdf(x))
                x_axis = "X_AXIS"
                y_axis = "Y_AXIS"
```

Código 5: Distribucion Binomial

El cálculo de la distribución poisson descansa en la clase abstracta y el uso del DataFrame de pandas

```
        elif tipo_distribucion == 'Poisson':

            datos = {}
            datos['mu'] = pois_valor_mu
            datos['x'] = pois_valor_x
            datos['acumulada'] = acumulada_switch

            poisson = d.getDistribution("Poisson",datos)

            # Crear DataFrame
            if not acumulada_switch:
                df = pd.DataFrame(poisson.get_sample(pois_valor_x),
columns=['n_gen'])
                df['pdf'] = df['n_gen'].apply(lambda x: poisson.
get_probability(x))
                x_axis = "X_AXIS"
                y_axis = "Y_AXIS"
            else:
                df = pd.DataFrame(poisson.get_sample(pois_valor_x),
columns=['n_gen'])
                df['pdf'] = df['n_gen'].apply(lambda x: poisson.
get_probability_cdf(x))
                x_axis = "X_AXIS"
                y_axis = "Y_AXIS"
```

Código 6: Distribucion Binomial

El cálculo de la distribución geometrica descansa en la clase abstracta y el uso del DataFrame de pandas

```
1
2         elif tipo_distribucion == 'Geometrica':
3
4             datos = {}
5             datos['p'] = geom_valor_p
6             datos['x'] = geom_valor_x
7             datos['acumulada'] = acumulada_switch
8
9             geometrica = d.getDistribution("Geometrica",datos)
10
11            # Crear DataFrame
12            if not acumulada_switch:
13                df = pd.DataFrame(geometrica.get_sample(geom_valor_x),
    columns=['n_gen'])
14                df['pdf'] = df['n_gen'].apply(lambda x: geometrica.
    get_probability(x))
15                x_axis = "X_AXIS"
16                y_axis = "Y_AXIS"
17            else:
18                df = pd.DataFrame(geometrica.get_sample(geom_valor_x),
    columns=['n_gen'])
19                df['pdf'] = df['n_gen'].apply(lambda x: geometrica.
    get_probability_cdf(x))
20                x_axis = "X_AXIS"
21                y_axis = "Y_AXIS"
```

Código 7: Distribucion Geometrica

El cálculo de la distribución exponencial descansa en la clase abstracta y el uso del DataFrame de pandas

```
        elif tipo_distribucion == 'Exponencial':

            datos = {}
            datos['alpha'] = exp_valor_alpha
            datos['x'] = exp_valor_x
            datos['acumulada'] = acumulada_switch

            exponencial = d.getDistribution("Exponencial",datos)

            if not acumulada_switch:
                df = pd.DataFrame(exponencial.get_sample(exp_valor_x),
    columns=['n_gen'])
                df['pdf'] = df['n_gen'].apply(lambda x: exponencial.
    get_probability(x))
                x_axis = "X_AXIS"
                y_axis = "Y_AXIS"
            else:
                df = pd.DataFrame(exponencial.get_sample(exp_valor_x),
    columns=['n_gen'])
                df['pdf'] = df['n_gen'].apply(lambda x: exponencial.
    get_probability_cdf(x))
                x_axis = "X_AXIS"
                y_axis = "Y_AXIS"
```

Código 8: Distribucion Geometrica

El cálculo de la distribución normal descansa en la clase abstracta y el uso del DataFrame de pandas, al terminar el if-else se crea el gráfico con la información generada

```
1
2         else:
3
4             datos = {}
5             datos['mu'] = float(norm_valor_mu)
6             datos['sigma'] = float(norm_valor_sigma)
7             datos['acumulada'] = acumulada_switch
8
9             # Crear objeto distribucion normal
10            normal = d.getDistribution("Normal",datos)
11
12            if not acumulada_switch:
13                df = pd.DataFrame(normal.get_sample(norm_valor_x),
    columns=['n_gen'])
14                df['pdf'] = df['n_gen'].apply(lambda x: normal.
    get_probability(x))
15     x_axis = "X_AXIS"
16                y_axis = "Y_AXIS"
17            else:
18                df = pd.DataFrame(normal.get_sample(norm_valor_x),
    columns=['n_gen'])
19                df['pdf'] = df['n_gen'].apply(lambda x: normal.
    get_probability_cdf(x))
20                x_axis = "X_AXIS"
21                y_axis = "Y_AXIS"
22
23        # Regresar grafico de respuesta
24        fig = go.Figure()
25        fig.add_trace(go.Scatter(
26            x=df['n_gen'],
27            y=df['pdf'],
28            mode="markers"
29        ))
30
31        fig.update_layout(
32            title=tipo_distribucion,
33            xaxis_title = x_axis,
34            yaxis_title = y_axis
35        )
36
37        return fig,texto(tipo_distribucion,acumulada_switch)
```

Código 9: Distribucion Geometrica

Correr el servidor

```
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5001, debug=False)
```

Código 10: Ejecución Server