Libraries

```
from plotly.offline import init_notebook_mode, iplot
py.init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.tools as tls
import warnings

from matplotlib import rcParams
import matplotlib.pyplot as plt
import seaborn as sns
import re
import string
from collections import Counter

import plotly.express as px
import plotly.figure_factory as ff
from IPython.display import HTML, Image
import plotly
import plotly.plotly as py
```

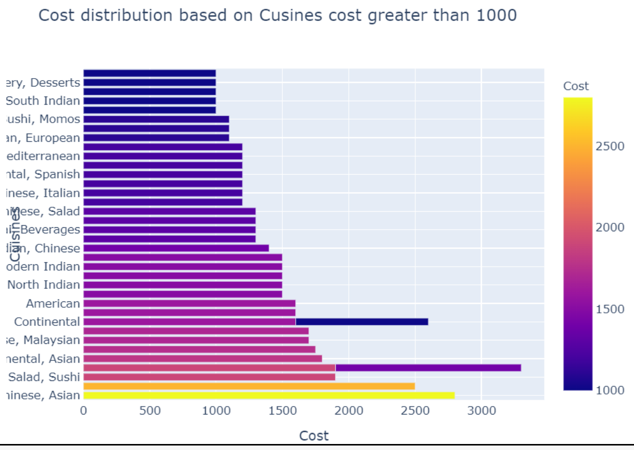
Cufflinks (Interactive Pandas Dataframe plotting)

```
#importing Pandas
import pandas as pd
#importing plotly and cufflinks in offline mode
import cufflinks as cf
import plotly.offline
cf.go_offline()
cf.set_config_file(offline=False, world_readable=True)
```

https://github.com/santosjorge/cufflinks/blob/master/Cufflinks%20Tutorial%20-%20Pandas%20Like.ipynb

Barplot

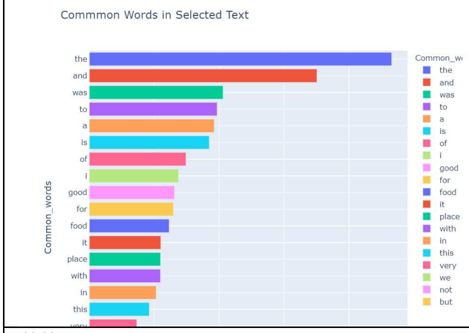
```
## simple barplot with plotly.express

fig = px.bar(cusine_sort_price, x='Cost', y='Cuisines', color='Cost',title="Cost distribution based on Cusines cost greater than 1000",)
fig.show( )
```



```
## barplot with plotly.graph_objs
data = [go.Bar(
            x = train_df["target"].value_counts( ).index.values,
            y = train_df["target"].value_counts( ).values,
            text='Distribution of target variable'
    )]

layout = go.Layout(
    title='Target variable distribution'
)

fig = go.Figure(data=data, layout=layout)

py.iplot(fig, filename='basic-bar')
```

## Horizontal barplot with plotly.express

```
fig = px.bar(temp, x="count", y="Common_words", title='Commmon Words in Selected Text',
orientation='h', width=700, height=700,color='Common_words')
fig.show()
```

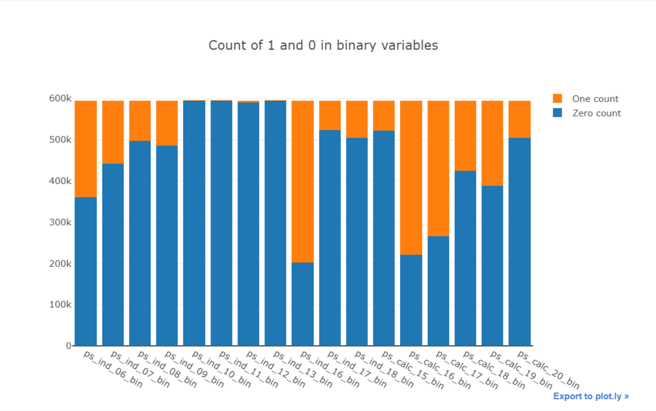


## Missing Value detecting visualization with barplot in Plotly

```
def mis_value_graph(data):
    data = [
    go.Bar(
        x = data.columns,
        y = data.isnull().sum(),
        name = 'Counts of Missing value',
        textfont=dict(size=20),
        marker=dict(
        line=dict(
            color= generate_color(),
            #width= 2,
        ), opacity = 0.45
    )
    ),
    ]
    layout= go.Layout(
        title= '"Total Missing Value By Column"',
        xaxis= dict(title='Columns', ticklen=5, zeroline=False, gridwidth=2),
        yaxis= dict(title='Value Count', ticklen=5, gridwidth=2),
        showlegend=True
    )
    fig = go.Figure(data=data, layout=layout)
    py.iplot(fig, filename='skin')

def generate_color():
    color = '#{:02x}{:02x}{:02x}'.format(*map(lambda x: random.randint(0, 255), range(3)))
    return color
```

## stacked barplot with plotly.graph_obj

```
bin_col = [col for col in train.columns if '_bin' in col]
zero_list = []
one_list = []
for col in bin_col:
        zero_list.append((train[col]==0).sum())
        one_list.append((train[col]==1).sum())

trace1 = go.Bar(
        x=bin_col,
        y=zero_list ,
        name='Zero count'
)
trace2 = go.Bar(
        x=bin_col,
        y=one_list,
        name='One count'
)

data = [trace1, trace2]
layout = go.Layout(
        barmode='stack',
        title='Count of 1 and 0 in binary variables'
)

fig = go.Figure(data=data, layout=layout)
py.iplot(fig, filename='stacked-bar')
```



## Forest Algorithms Feature Importance Bar Plots with Plotly

```
x, y = (list(x) for x in zip(*sorted(zip(rf.feature_importances_, features),
                                                                        reverse = False)))
trace2 = go.Bar(
        x=x ,
        y=y,
        marker=dict(
                color=x,
                colorscale = 'Viridis',
                reversescale = True
        ),
        name='Random Forest Feature importance',
        orientation='h',
)

layout = dict(
        title='Barplot of Feature importances',
          width = 900, height = 2000,
        yaxis=dict(
                showgrid=False,
                showline=False,
                showticklabels=True,
#                 domain=[0, 0.85],
        ))

fig1 = go.Figure(data=[trace2])
fig1['layout'].update(layout)
```

```
py.iplot(fig1, filename='plots')
```

Line Plots with dot markers

```
#using plotly Scatter
time_plot_1=go.Figure(go.Scatter(x=time_plot_1_df.Year, y=time_plot_1_df.Total_Fires,
                                                        mode='lines+markers', line={'color': 'red'}))
#layout changes
time_plot_1.update_layout(title='Brazil Fires per 1998-2017 Years',
                                xaxis_title='Year',
                                yaxis_title='Fires')
#showing the figure
time_plot_1.show( )
```

Pie Graph

```
# rcParams["figure.figsize"]    =15,10
# restarant_names["Rating"].value_counts().plot(kind="pie")
fig = px.pie(values=restarant_names["Rating"].value_counts(),hover_name=[5.0,4.0,3.0,2.0,1.0],title="Hotel Rating Distribution")
fig.show()
```

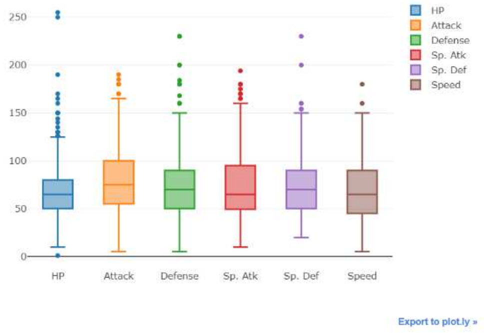
Distribution Plot / Histogram / BoxPlot

## Plotly “create_displot”

```
from plotly import tools
import plotly.plotly as py
from plotly.offline import init_notebook_mode, iplot
init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.figure_factory as ff

fig = ff.create_distplot([some array],['name of  legend label'],bin_size=0.001)
iplot(fig, filename='Basic Distplot')
```

## Boxplot with go

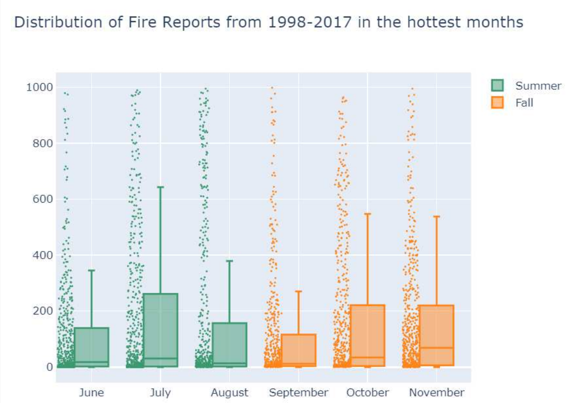
```
trace0 = go.Box(y=df["HP"],name="HP")
trace1 = go.Box(y=df["Attack"],name="Attack")
trace2 = go.Box(y=df["Defense"],name="Defense")
trace3 = go.Box(y=df["Sp. Atk"],name="Sp. Atk")
trace4 = go.Box(y=df["Sp. Def"],name="Sp. Def")
trace5 = go.Box(y=df["Speed"],name="Speed")
data = [trace0, trace1, trace2,trace3, trace4, trace5]
iplot(data)
```



## Boxplot with px

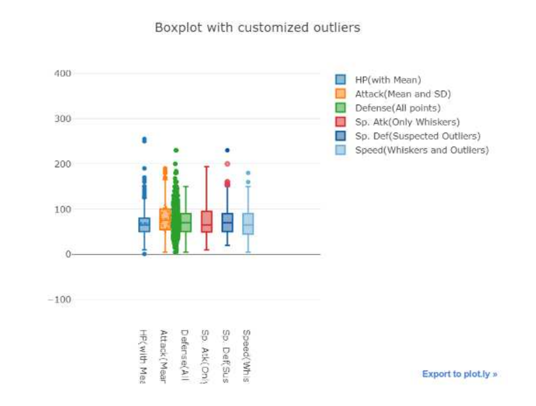
```
#isolating the hottest months by season
month_array_summer=['June','July','August']
month_array_fall=['September','October','November']
#leaving data only for hottest months
box_plot_df_summer=amazon_df.loc[amazon_df['Month'].isin(month_array_summer)]
box_plot_df_fall=amazon_df.loc[amazon_df['Month'].isin(month_array_fall)]
#visualizing reports
box_plot=go.Figure()
```

```
box_plot.add_trace(go.Box(y=box_plot_df_summer.Fire_Number, x=box_plot_df_summer.Month,
                                            name='Summer', marker_color='#3D9970',
                                            boxpoints='all', jitter=0.5, whiskerwidth=0.2,
                                            marker_size=2,line_width=2))
box_plot.add_trace(go.Box(y=box_plot_df_fall.Fire_Number, x=box_plot_df_fall.Month,
                                          name='Fall', marker_color='#FF851B',
                                          boxpoints='all', jitter=0.5, whiskerwidth=0.2,
                                            marker_size=2,line_width=2))

box_plot.update_layout(
            title_text = 'Distribution of Fire Reports from 1998-2017 in the hottest months')
box_plot.show()
```



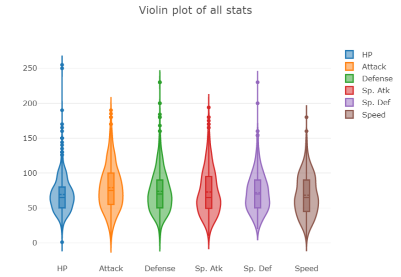
## Customized Boxplots

```
trace0 = go.Box(
      y=df["HP"],
      boxmean = True,
      name="HP(with Mean)"
)
trace1 = go.Box(
      y=df["Attack"],
      boxmean = 'sd',
      name="Attack(Mean and SD)"
)
trace2 = go.Box(
      y=df["Defense"],
      jitter = 0.5,
      pointpos = -2,
      boxpoints = 'all',
      name = "Defense(All points)"
)
trace3 = go.Box(
      y=df["Sp. Atk"],
      boxpoints = False,
      name = "Sp. Atk(Only Whiskers)"
)
trace4 = go.Box(
      y=df["Sp. Def"],
      boxpoints = 'suspectedoutliers',
      marker = dict(
            outliercolor = 'rgba(219, 64, 82, 0.6)',
            line = dict(
                  outliercolor = 'rgba(219, 64, 82, 0.6)',
                    outlierwidth = 2)),
        line = dict(
            color = 'rgb(8,81,156)'),
        name = "Sp. Def(Suspected Outliers)"
)
trace5 = go.Box(
        y=df["Speed"],
        boxpoints = 'outliers',
        line = dict(
            color = 'rgb(107,174,214)'),
        name = "Speed(Whiskers and Outliers)"
)

layout = go.Layout(
        title = "Boxplot with customized outliers"
)
data = [trace0, trace1, trace2, trace3, trace4, trace5]
fig = go.Figure(data=data,layout=layout)
iplot(fig, filename = "Customized Boxplot")
```



## Violin Plot

```
data = []
for i in range(5,11):
        trace = {
                    "type": 'violin',
                    "x": max(df.iloc[:,i]),
                    "y": df.iloc[:,i],
                    "name": list(df.columns)[i],
                    "box": {
                        "visible": True
                    },
                    "meanline": {
                        "visible": True
                    }
            }
        data.append(trace)

fig = {
        "data": data,
        "layout" : {
            "title": "Violin plot of all stats",
            "yaxis": {
                "zeroline": False,
            }
        }
}

iplot(fig, filename='violin', validate = False)
```



Scatter Plot

```
## Scatterplot with plotly.graph_objs

import plotly.offline as py
import plotly.graph_objs as go
py.init_notebook_mode(connected=True)
from plotly import tools
import plotly.figure_factory as ff
iris = datasets.load_iris()
X = iris.data[:, :2]    # we only take the first two features.
Y = iris.target

x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
trace = go.Scatter(x=X[:, 0],
                   y=X[:, 1],
                   mode='markers',
                   marker=dict(color=np.random.randn(150),
                               size=10,
                               colorscale='Viridis',
                               showscale=False))

layout = go.Layout(title='Training Points',
                   xaxis=dict(title='Sepal length',
                              showgrid=False),
                   yaxis=dict(title='Sepal width',
                              showgrid=False),
                  )

fig = go.Figure(data=[trace], layout=layout)
py.iplot(fig)
```

```
## RF Feature importance Plotly Scatter plot

trace = go.Scatter(
        y = rf.feature_importances_,
        x = features,
        mode='markers',
        marker=dict(
                sizemode = 'diameter',
                sizeref = 1,
                size = 13,
                #size= rf.feature_importances_,
                #color = np.random.randn(500), #set color equal to a variable
                color = rf.feature_importances_,
                colorscale='Portland',
                showscale=True
        ),
        text = features
)
data = [trace]

layout= go.Layout(
        autosize= True,
        title= 'Random Forest Feature Importance',
        hovermode= 'closest',
          xaxis= dict(
                  ticklen= 5,
                  showgrid=False,
                zeroline=False,
                showline=False
          ),
        yaxis=dict(
                title= 'Feature Importance',
                showgrid=False,
                zeroline=False,
                ticklen= 5,
                gridwidth= 2
        ),
        showlegend= False
)
fig = go.Figure(data=data, layout=layout)
py.iplot(fig,filename='scatter2010')
```

```
## Another scatter plot with plotly.graph_objs with diff design

trace1 = go.Scatter(
        x = df["Defense"],
        y = df["Attack"],
        mode='markers',
        marker=dict(
                size='16',
                color = df["Speed"],#set color equal to a variable
                colorscale='Electric',
                showscale=True
        ),
        text=df["Name"]
)
data = [trace1]
layout = go.Layout(
    paper_bgcolor='rgba(0,0,0,1)',
    plot_bgcolor='rgba(0,0,0,1)',
    showlegend = False,
    font=dict(family='Courier New, monospace', size=10, color='#ffffff'),
    title="Scatter plot of Defense vs Attack with Speed as colorscale",
)
fig = go.Figure(data=data, layout=layout)
iplot(fig, filename = "Scatterplot")
```

```
# 3D Scatter Plot in Plotly

trace = go.Scatter3d(x=X_reduced[:, 0],
                                          y=X_reduced[:, 1],
                                          z=X_reduced[:, 2],
                                          mode='markers',
                                          marker=dict(
                                                  size=6,
                                                  color=np.random.randn(150),
                                                  colorscale='Viridis',
                                                  opacity=0.8)
                                        )
layout=go.Layout(title='First three PCA directions',
                                  scene=dict(
                                                  xaxis=dict(title='1st eigenvector'),
                                                  yaxis=dict(title='2nd eigenvector'),
                                                  zaxis=dict(title='3rd eigenvector'))
                                  )
fig = go.Figure(data=[trace], layout=layout)
py.iplot(fig)
```

```
# 3D Scatter Plot in Plotly 2

trace = go.Scatter3d(
        x = data['chol'],
        y = data['trestbps'],
        z = data['age'],
        name = 'Marvel',
        mode = 'markers',
        marker = dict(
                  size = 10,
                  color = data['age']
        )
)

df = [trace]

layout = go.Layout(
        title = 'Cholestrol vs Heart Rate vs Age',
        margin=dict(
                l=0,
                r=0,
                b=0,
                t=0
        ),
        scene = dict(
                        xaxis = dict(title    = 'Cholestrol'),
                        yaxis = dict(title    = 'Heart Rate'),
                        zaxis = dict(title    = 'Age')
                )

)
fig = go.Figure(data = df, layout=layout)
py.iplot(fig)
```

Heatmap

```
# Heatmap with Plotly

data = [
        go.Heatmap(
                z= train_int.corr().values,
                x= train_int.columns.values,
                y= train_int.columns.values,
                colorscale='Viridis',
                reversescale = False,
                #text = True ,
                opacity = 1.0 )
]

layout = go.Layout(
        title='Pearson Correlation of Integer-type features',
        xaxis = dict(ticks='', nticks=36),
        yaxis = dict(ticks='' ),
        width = 900, height = 700)

fig = go.Figure(data=data, layout=layout)
py.iplot(fig, filename='labelled-heatmap')
```

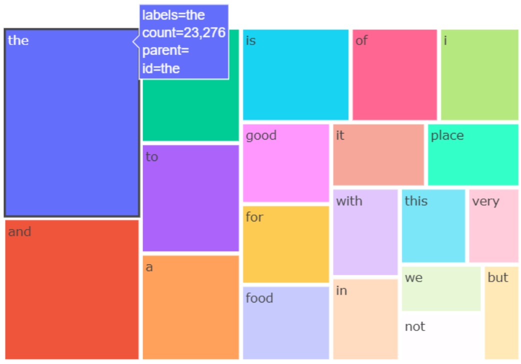
BubblePlot

```
!pip install bubbly

import matplotlib.pyplot as plt
import seaborn as sns

# for advanced visualizations
import plotly.offline as py
from plotly.offline import init_notebook_mode, iplot
import plotly.graph_objs as go
init_notebook_mode(connected = True)
```

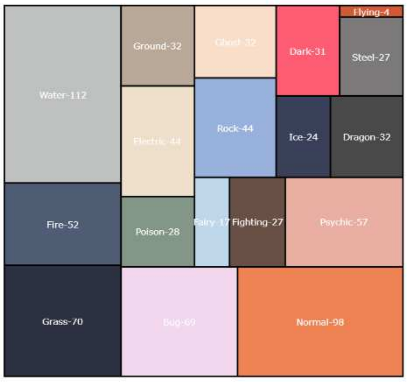
```
from bubbly.bubbly import bubbleplot

import warnings
warnings.filterwarnings('ignore')

figure = bubbleplot(dataset = data, x_column = 'trestbps', y_column = 'chol',
        bubble_column = 'sex', time_column = 'age', size_column = 'oldpeak', color_column = 'sex',
        x_title = "Resting Blood Pressure", y_title = "Cholestrol", title = 'BP vs Chol. vs Age vs Sex vs Heart
Rate',
        x_logscale = False, scale_bubble = 3, height = 650)

py.iplot(figure, config={'scrollzoom': True})
```

TreeMap

```
fig = px.treemap(temp, path=['Common_words'], values='count',title='Tree of Most Common Words')
fig.show()
```



```
import squarify

x = 0.
y = 0.
width = 50.
height = 50.
type_list = list(df["Type 1"].unique())
values = [len(df[df["Type 1"] == i]) for i in type_list]

normed = squarify.normalize_sizes(values, width, height)
rects = squarify.squarify(normed, x, y, width, height)

# Choose colors from http://colorbrewer2.org/ under "Export"
color_brewer =
['#2D3142','#4F5D75','#BFC0C0','#F2D7EE','#EF8354','#839788','#EEE0CB','#BAA898','#BFD7EA','#6
85044','#E9AFA3','#99B2DD','#F9DEC9','#3A405A','#494949','#FF5D73','#7C7A7A','#CF5C36','#EFC88
B']
shapes = []
annotations = []
counter = 0

for r in rects:
        shapes.append(
                dict(
                        type = 'rect',
                        x0 = r['x'],
                        y0 = r['y'],
                        x1 = r['x']+r['dx'],
                        y1 = r['y']+r['dy'],
            line = dict( width = 2 ),
            fillcolor = color_brewer[counter]
        )
    )
    annotations.append(
        dict(
            x = r['x']+(r['dx']/2),
            y = r['y']+(r['dy']/2),
            text = "{}-{}".format(type_list[counter], values[counter]),
            showarrow = False
        )
    )
    counter = counter + 1
    if counter >= len(color_brewer):
        counter = 0

# For hover text
trace0 = go.Scatter(
    x = [ r['x']+(r['dx']/2) for r in rects ],
    y = [ r['y']+(r['dy']/2) for r in rects ],
    text = [ str(v) for v in values ],
    mode = 'text',
)

layout = dict(
    height=700,
    width=700,
    xaxis=dict(showgrid=False,zeroline=False),
    yaxis=dict(showgrid=False,zeroline=False),
    shapes=shapes,
    annotations=annotations,
    hovermode='closest',
    font=dict(color="#FFFFFF")
)

# With hovertext
figure = dict(data=[trace0], layout=layout)
iplot(figure, filename='squarify-treemap')
```



Radar Plot

```
x = df[df["Name"] == "Charizard"]
data = [go.Scatterpolar(
  r = [x['HP'].values[0],x['Attack'].values[0],x['Defense'].values[0],x['Sp. Atk'].values[0],x['Sp.
Def'].values[0],x['Speed'].values[0],x["HP"].values[0]],
  theta = ['HP','Attack','Defense','Sp. Atk','Sp. Def','Speed','HP'],
  fill = 'toself'
)]
```

```
layout = go.Layout(
    polar = dict(
        radialaxis = dict(
            visible = True,
            range = [0, 250]
        )
    ),
    showlegend = False,
    title = "Stats of {}".format(x.Name.values[0])
)
fig = go.Figure(data=data, layout=layout)
iplot(fig, filename = "Single Pokemon stats")
```

```
# Creating a method to compare 2 pokemon
def compare2pokemon(x,y):
        x = df[df["Name"] == x]
        y = df[df["Name"] == y]

        trace0 = go.Scatterpolar(
            r = [x['HP'].values[0],x['Attack'].values[0],x['Defense'].values[0],x['Sp. Atk'].values[0],x['Sp.
Def'].values[0],x['Speed'].values[0],x["HP"].values[0]],
            theta = ['HP','Attack','Defense','Sp. Atk','Sp. Def','Speed','HP'],
            fill = 'toself',
            name = x.Name.values[0]
        )

        trace1 = go.Scatterpolar(
            r = [y['HP'].values[0],y['Attack'].values[0],y['Defense'].values[0],y['Sp. Atk'].values[0],y['Sp.
Def'].values[0],y['Speed'].values[0],y["HP"].values[0]],
            theta = ['HP','Attack','Defense','Sp. Atk','Sp. Def','Speed','HP'],
            fill = 'toself',
            name = y.Name.values[0]
        )

        data = [trace0, trace1]

        layout = go.Layout(
            polar = dict(
                radialaxis = dict(
                    visible = True,
                    range = [0, 200]
                )
            ),
            showlegend = True,
            title = "{} vs {}".format(x.Name.values[0],y.Name.values[0])
        )
        fig = go.Figure(data=data, layout=layout)
        iplot(fig, filename = "Two Pokemon stats")

# Comparing primeape and muk
compare2pokemon("Primeape","Muk")
```

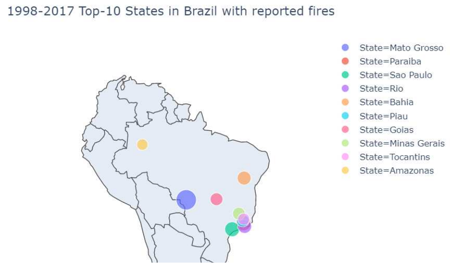
Scatter Plot Matrix with boxplots on diagonal line

```
fig = ff.create_scatterplotmatrix(df.iloc[:,5:12], index='Generation', diag='box', size=2, height=800,
width=800)
```

```
iplot(fig, filename ='Scatterplotmatrix.png',image='png')
```

Scatter Plot on Geographical Map

```
#using scatter geo with above created subdataframe
fig = px.scatter_geo(data_frame=geo_plot_df, scope='south america',lat='Lat',lon='Long',
                                        size='Count', color='State', projection='hammer')
fig.update_layout(
               title_text = '1998-2017 Top-10 States in Brazil with reported fires')
fig.show()
```


1998-2017 Top-10 States in Brazil with reported fires
● State=Mato Grosso
● State=Paraiba
● State=Sao Paulo
● State=Rio
● State=Bahia
● State=Piau
● State=Goias
● State=Minas Gerais
● State=Tocantins
● State=Amazonas