Plotly Document

1. Visualization plot by Feature type

1.1 Univariate

1. Quantitive features
   - Histograms
   - Density plot
   - Box plot
   - Violin plot

1. Categorical features(Discrete features)
   - Frequency table
   - Bar plot(Count)

1.2 Multivariate

1. Quantitive vs Quantitive
   - Correlation matrix
   - Scatterplot
   - Covariance_matrix
   - 1 + Categorical : scatter plot with hue

1. Quantitive vs Categorical
   - Box plot
   - 2 + Categorical : Box plot with subplots

1. Categorical vs Categorical
   - Bar plot(Count)

2. Heatmap

```
corr = df_cvasi.new.corr()
mask = np.triu(np.ones_like(corr, dtype=bool))
corr = corr.mask(mask)

fig = go.Figure()
fig.add_trace(go.Heatmap(
        z = corr,
        x = corr.columns.tolist(),
        y = corr.columns.tolist(),
        colorscale = 'RdBu',
        xgap = 1,
        ygap = 1,
        hoverinfo = "none"
    )
)

fig.update_layout(
    {
        "title": {
            "text": "<b>Correlation in each feaatures</b>",
            "x": 0.5,
            "y": 0.9,
            "font": {
                "size": 15
            }
        },
        "xaxis": {
            "title": "Columns",
            "tickfont": {
                "size": 8
            }
        },
        "yaxis": {
            "title": "Columns",
            "tickfont": {
                "size": 8
            }
        },
        "template":'plotly_white',
        "yaxis_autorange" : "reversed"
    }
)
fig.show()


# triangle Heatmap

corr = df_new.corr()
mask = np.triu(np.ones_like(corr, dtype=bool))
df_mask = corr.mask(mask)

fig = ff.create_annotated_heatmap(z=df_mask.to_numpy(),
                                  x=df_mask.columns.tolist(),
                                  y=df_mask.columns.tolist(),
                                  colorscale=px.colors.diverging.RdBu,
                                  hoverinfo="none",
                                  showscale=True, ygap=1, xgap=1
                                 )

fig.update_xaxes(side="bottom")

fig.update_layout(
    title_text='Heatmap',
    title_x=0.5,
    width=1000,
    height=1000,
    xaxis_showgrid=False,
    yaxis_showgrid=False,
    xaxis_zeroline=False,
    yaxis_zeroline=False,
    yaxis_autorange='reversed',
    template='plotly_white'
)

for i in range(len(fig.layout.annotations)):
    if fig.layout.annotations[i].text == 'nan':
        fig.layout.annotations[i].text = ""
    fig.layout.annotations[i].font.size = 8

fig.show()
```

3. Scatter_Matrix

Plot px.scatter_matrix show correlation of each other features with scatter points.

```
fig = px.scatter_matrix(cancer_df, dimensions =['mean radius', 'mean area', 'mean perimeter', 'mean texture'],
    color = 'target', width = 1000, height = 1000)
fig.show()
```

4. Make dataset for visualization

4.1 melt

```
df_uniques = pd.melt(
    df_cvasi,
    value_vars=['gender', "cholesterol", "gluc", "smoke", "alco", "active", "cardio"],
)
df_uniques = (
    pd.DataFrame(df_uniques.groupby(["variable", "value"])["value"].count())
        .sort_index(level=[0, 1])
        .rename(columns={"value": "count"})
        .reset_index()
)

sns.factorplot(
    x="variable", y="count", hue="value", data=df_uniques, kind="bar", size=12
);
```

Difference between go.Figure and fig.add_trace()

URL : Difference between go.object and add_trace method :

5. iplot

5.1 layout

```
layout = {
    'title' :
    {
        'text' : '<b>Test Graph for EDA</b>',
        'font' : {
            'size' : 35,
            'color' : '#374748'
        },
        'x' : 0.5,
        'y' : 0.88
    },
    'plot_bgcolor' : '#FAFAFA',
    'xaxis' : {
        'showticklabels' : True,
        'dtick' : 1,
        'title' : {
            'text' : 'Axis1',
            'font' : {
                'size' : 30,
                'color' : '#374748'
            }
        }
    },
    'yaxis' : {
        'showticklabels' : True,
        'dtick' : 2,
        'title' : {
            'text' : 'Axis1',
            'font' : {
                'size' : 30,
                'color' : '#374748'
                }

            }

        }
    }
```

6. plotly.graph_obj()

6.1 layout

```
fig.update_layout(
    {
        "title": {
            "text": "<b>Percentage or heart rate by Age decades and sex</b>",
            "x": 0.5,
            "y": 0.9,
            "font": {
                "size": 15
            }
        },
        "xaxis": {
            "title": "Age Decades",
            "showticklabels":True,
            "tickfont": {
                "size": 10
            }
        },
        "yaxis": {
            "title": "Percentage of HeartDisease",
            "tickfont": {
                "size": 10
            }
        },
        "template":'plotly_white'
    }
)
```

6.2 annotation

```
fig.add_annotation(
            x='2017-11-30',
            y=0.85,
            text="<b>Modest Monthly Turnovers</b>",
            showarrow=True,
            font=dict(
                size=15,
                color="#ffffff"
                ),
            align="center",
            arrowhead=2,
            arrowsize=1,
            arrowwidth=2,
            arrowcolor="#7fCFD9",
            ax=25,
            ay=-30,
            bordercolor="#7fCFD9",
            borderwidth=2,
            borderpad=4,
            bgcolor="#F2D5D8",
            opacity=1.0
)
```

6.3 hue of plot

```
# Plotly same as seaborn hue
# If we use groupby option in Data Frame :
# region becomes index of gourpby index and geo_region becomes Data Frame by index

traces = []
for region, geo_region in geo_groupby('Geographical region'):
    traces.append(go.Scatter(x=geo_region.Year, y=geo_region.Number, name=region, mode='lines'))

fig = go.Figure(data=traces)
```

7. Subplots

```
def create_count_plot(fea) :
    grouped_df = train_df.groupby(fea).size().reset_index()
    grouped_df.columns = [fea, 'Count']

    grouped_df_target = train_df.groupby([fea, 'Transported']).size().reset_index()
    grouped_df_target.columns = [fea, 'Transported', 'Count']

    fig = make_subplots(rows=1, cols=2)

    fig.add_trace(go.Bar(
        x = grouped_df[fea],
        y = grouped_df['Count'],
    ), row = 1, col = 1)

    for trans in train_df['Transported'].unique() :
        plot_df = grouped_df_target[grouped_df_target['Transported'] == trans]
        fig.add_trace(go.Bar(
            x = plot_df[fea],
            y = plot_df['Count'],
            name = 'Transported (trans)'
        ), row = 1, col = 2)

    fig.update_layout(
    {
        "title": {
            "text": f"Countplots of {fea}",
            "x": 0.5,
            "y": 0.9,
            "font": {
                "size": 15
            }
        },
        "yaxis": {
            "title": "Count",
            "tickfont": {
                "size": 10
            }
        },
        "template":'plotly_dark'
    }
    )

    fig.update_xaxes(title_text=fea, row=1, col=1)
    fig.update_xaxes(title_text=fea, row=1, col=2)

    fig.show()
```

8. Colors

8.1 specific color by value

```
color = ['#D5DBDC',] * len(df_city.index)
color[3] = '#F2A472'
color[7] = '#F2A472'
```

8.2 whole color scale

```
px.colors.sequential.RdBu
```

8.3 specific color by condition

```
fig = go.Figure()

fig.add_trace(go.Bar(
    x = df[df.Name == 'Counter-Strike'].Datetime,
    y = df[df.Name == 'Counter-Strike'].Gain,
    marker_color =
        color = pd.Series(np.where(df[df.Name == 'Counter-Strike'].Gain == 0, 'pos', 'neg')).astype(str).map({
    'pos' : 'blue', 'neg' : 'red'})
))

fig.show()
```

9. Discrete plot by value

```
colors = px.colors.sequential.RdBu[::11]
for month in df_attacker['Month'].unique() :
    if month != 0 :
        fig.add_trace(
            go.Bar(
                x = df_attacker.loc[df_attacker.Month == month, 'Age_decade'],
                y = df_attacker.loc[df_attacker.Month == month, 'Critical'],
                text = df_attacker.loc[df_attacker.Month == month, 'Critical'],
                textposition = "outside",
                marker_color = colors
            )
        )

    fig.update_layout(
    {
        "title": {
            "text": "<b>Critical Proportion in months</b>",   # Can add title value using f" {}"
            "x": 0.5,
            "y": 0.9,
            "font": {
                "size": 15
            }
        },
        "xaxis": {
            "title": "Age_decade from <20 to 65~",
            "showticklabels":True,
            "tickfont": {
                "size": 10
            }
        },
        "yaxis": {
            "title": "Critical",
            "tickfont": {
                "size": 10
            }
        },
        "template":'plotly_white'
    }
    )

    fig.show()
```

10. Use unique vlaue

```
fig = go.Figure()

for loc in Location_index :
    fig.add_trace(go.Scatter(
        x = df_loc['Year'].unique(),
        y = df_loc.loc[df_loc['Location'] == loc, 'Count'],
        text = df_loc.loc[df_loc['Location'] == loc, 'Count'],
        name = loc))

fig.update_layout(
    {
        "title": {
            "text": "<b>Count of crime by Sex</b>",
            "x": 0.5,
            "y": 0.9,
            "font": {
                "size": 15
            }
        },
        "xaxis": {
            "title" : "Year",
            "dtick" : "1",
            "tickfont": {
                "size": 10
            }
        },
        "yaxis": {
            "title": "Count of Crime",
            "tickfont": {
                "size": 10
            }
        },
        "template":'plotly_white'
    }
)
fig.show()
```