Plotly Graph Objects Membantu Visualisasi Data yang Interaktif dan Menarik

Plotly adalah sebuah pustaka (*library*) salah satunya dalam bahasa *Python* untuk visualisasi data yang memungkinkan pengguna untuk membuat grafik interaktif dan menarik. Pustaka ini sangat populer di kalangan ilmuwan data dan analis karena kemampuannya untuk menghasilkan visualisasi yang mudah dibagikan dan diintegrasikan ke dalam aplikasi web.

Gambaran Umum *Plotly*

Plotly memiliki beberapa modul yang dirancang untuk berbagai tujuan dan jenis visualisasi. Berikut ini beberapa modul utama *Plotly*:

1. ***plotly.graph_objects***: Modul ini digunakan untuk membuat dan mengonfigurasi plot secara mendetail dengan pendekatan berorientasi objek. Pengguna dapat mengontrol setiap aspek visualisasi, termasuk data, tata letak, dan gaya.
2. ***plotly.express****:* Menyediakan cara yang lebih sederhana dan cepat untuk membuat visualisasi. Cocok untuk eksplorasi data dan pembuatan plot dasar dengan sintaks yang lebih ringkas.
3. ***plotly.subplots****:* Memungkinkan pembuatan subplot, yaitu beberapa plot dalam satu *figure*, yang sangat berguna untuk membandingkan beberapa dataset atau jenis visualisasi.
4. ***plotly.io****:* Digunakan untuk fungsi input dan output, seperti menyimpan plot ke file atau menampilkannya dalam format tertentu.
5. ***plotly.colors****:* Menyediakan fungsionalitas terkait warna, termasuk palet warna dan skema warna yang dapat digunakan dalam visualisasi.
6. ***plotly.graph_objs****:* Modul ini mirip dengan *graph_objects*, biasanya digunakan untuk mendefinisikan objek-objek plot yang lebih kompleks dan spesifik.
7. ***plotly.figure_factory****:* Memudahkan pembuatan *figure* dengan cara yang lebih tinggi, seperti heatmaps, violin plots, dan visualisasi lainnya yang memerlukan lebih sedikit konfigurasi manual.

Setelah mendapat gambaran umum dari pustaka *Plotly* yang memiliki modul cukup beragam, artikel ini akan membahas secara khusus mengenai *Plotly Graph Objects*. Sesuai dengan fungsi utama dari modul ini yang sudah disinggung sebelumnya, yaitu untuk membuat, memanipulasi, dan merender visualisasi, modul ini juga dapat digunakan untuk mengonfigurasi plot secara mendetail dengan pendekatan berorientasi objek. Pengguna dapat mengontrol setiap aspek visualisasi, termasuk data, tata letak, dan gaya.

Instalasi *Plotly* dan Impor *Plotly Graph Objects*

Apabila *Plotly* belum disertakan dalam *Python*, pustaka ini dapat diinstalasi terlebih dahulu pada terminal.

```
pip install plotly
```

Setelah instalasi berhasil dilakukan, maka tahapan selanjutnya adalah mengimpor modul *graph objects* sebagai berikut:

```
import plotly.graph_objects as go
```

Kelas utama dari modul ini adalah *Figure* dan varian *ipywidgets* yang kompatibel yaitu *FigureWidget*. Keduanya merepresentasikan keseluruhan dari visualisasi atau plot. Pada *Figure* dan *FigureWidget* memungkinkan pengguna untuk memanipulasi atribut-atribut seperti `.update_layout()` atau `.add_trace()`, di mana kesemua perintahnya menggunakan notasi *underscore* sebagaimana jika merendernya misalnya `.show()` ataupun melakukan ekspor ke berbagai format seperti `.to_json()` atau `.write_image()` atau `.write_html()`.

Atribut *non-leaf* pada suatu *figure* direpresentasikan dari suatu kelas di dalam hirarki *plotly.graph_objects*. Misalnya, suatu objek atau *Figure* yang telah dibuat dan disimpan dalam sebuah variable bernama *fig* (yang umum digunakan untuk memudahkan pemahaman penamaan objek dimaksud). Selanjutnya, dari variable objek bernama *fig* tersebut memiliki atribut di antaranya `layout.margin`, yang berisi atribut-atribut `t` (margin atas / *top*), `l` (margin kiri / *left*), `b` (margin bawah / *bottom*) dan `r` (margin kanan / *right*) . Misalnya `fig.layout,margin`, di mana pengguna bisa mengatur margin yang diinginkan. Selain itu, nilai-nilai pada atribut t,l,b dan r dapat diatur dengan menggunakan ***magic underscore notation***. *Magic underscore notation* merupakan cara penulisan yang memungkinkan pengguna untuk mengatur atribut dari objek secara langsung, menggunakan notasi yang lebih ringkas.

Pengguna dapat membuat objek *Figure* dan langsung mengatur atribut layout_margin tanpa perlu membuat objek *layout* atau margin terlebih dahulu, misalnya:

```
layout = go.Layout(margin=go.layout.Margin(t=10, b=10, r=10, l=10))
fig = go.Figure(layout=layout)
```

Namun, dengan menggunakan *magic underscore notation*, dapat langsung menetapkan nilai margin dalam satu langkah yang lebih ringkas dan muda dibaca.

Objek yang merepresentasikan suatu *dataset* yang akan ditampilkan dalam visualisasi disebut juga *traces.* Terdapat lebih dari 40 tipe *traces* dalam pustaka *Plotly*, seperti garis, titik, batang, permukaan dan lain-lain. Setiap tipe *trace* memiliki kelasnya sendiri di modul *plotly.graph_objects*. Pengguna dapat memilih tipe *traces* yang paling sesuai untuk data yang ini divisualisasi agar mudah dibaca. Selanjutnya akan lebih jelas dibahas pada studi kasus di artikel ini.

Perintah umum pada Plotly Graph Objects (`.go`)

Berikut perintah yang umum digunakan pada *Plotly Graph Objects*.

- `go.Figure()` : Pembuataan *figure.*
- `add_traces()` : Menambahkan *traces* / data series ke dalam *figure.*
- `Scatter()`: Membuat *trace* diagram sebar.
- `Bar()` : Membuat *trace* grafik batang.
- `Histogram()` : Membuat *trace* histogram.
- `Box()` : Membuat *trace* diagram kotak.
- `Heatmap()` : Membuat *trace heatmap*.

- `Surface()`: Membuat *trace* grafik permukaan 3D.
- `update_layout()`: Memperbaharui properti *layout.*
- `title(), xaxis(), yaxis(), legend(), coloraxis()`: Mengatur atribut.
- `update_traces(), marker(), line(), fill()`: Mengkustomisasi / mengatur *trace.*
- `add_annotation(), add_shape()`: Menambahkan anotasi dan *shapes* atau bentuk geometris ke plot.
- `hovermode(), clickmode(), on_hover(), on_click()`: interaktivitas (mengatur interaksi data).
- `write_html(), show()`: Menyimpan *figure* sebagai file HTML atau menampilkannya.

Kegunaan dan Pengaplikasian

Kegunaan dari *Plotly Graph Objects* adalah sebagai berikut:

1. **Visualisasi Interaktif:** Memungkinkan pembuatan visualisasi interaktif dengan efek hover, peristiwa klik, dan kemampuan *zoom*.
2. **Kustomisasi:** Menyediakan berbagai opsi untuk kustomisasi, termasuk pengaturan layout dan gaya.
3. **Variasi Tipe Plot:** Mendukung berbagai jenis plot yang memungkinkan representasi data yang beragam.
4. **Integrasi yang Mulus:** Terintegrasi dengan *Jupyter Notebooks* dan dapat digunakan dalam skrip *Python* mandiri.
5. **Ekspor dan Berbagi:** Memungkinkan visualisasi diekspor ke format HTML atau gambar statis.

Plotly Graph Objects dapat diaplikasikan pada tahapan di antaranya:

1. **Eksplorasi dan Analisis Data:** Digunakan untuk analisis data eksploratif.
2. **Visualisasi Ilmiah:** Digunakan dalam bidang ilmiah untuk memvisualisasikan hasil eksperimen.
3. **Intelijen Bisnis:** Membantu membuat dasbor interaktif dan laporan yang menarik.
4. **Dasbor Data:** Digunakan untuk memantau indikator kinerja utama dan analisis data secara *real-time*.
5. ***Machine Learning* dan Ilmu Data:** Memvisualisasikan kinerja model dan aspek analitis lainnya.

Berikut adalah beberapa jenis plot yang bisa dibuat dengan *Plotly Graph Objects*:

- **Scatter Plot**: Memvisualisasikan data dalam bentuk titik-titik.
- **Line Plot**: Menghubungkan titik data dengan garis.
- **Bar Chart**: Menampilkan data dalam bentuk batang vertikal atau horizontal.
- **Histogram**: Menggambarkan distribusi frekuensi data.
- **Box Plot**: Menampilkan ringkasan statistik data, seperti median dan kuartil.
- **Heatmap**: Menunjukkan nilai data dalam format matriks dengan warna.
- **Pie Chart**: Menampilkan proporsi dari total dalam bentuk irisan.
- **Area Chart**: Mirip dengan line plot, tetapi area di bawah garis diwarnai.
- **Bubble Chart**: Scatter plot dengan ukuran titik yang mewakili nilai tambahan.
- **Surface Plot**: Visualisasi data tiga dimensi dengan permukaan halus.
- **3D Scatter Plot**: Scatter plot dalam tiga dimensi.
- **Funnel Chart**: Menunjukkan alur proses atau konversi.

Studi Kasus Plotly Graph Objects pada Dataset "*insurance*"

Pada artikel ini, mencoba penerapan *Plotly Graph Objects* pada sebuah data *insurance* yang berisi kolom sebagai berikut:

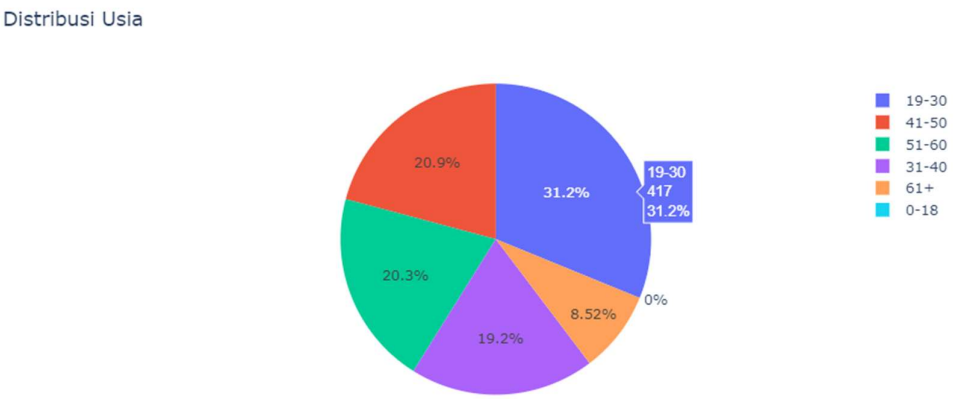
```
df = pd.read_csv('insurance.csv')
df.head()
```
✓ 0.1s Open 'df' in Data Wrangler

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

Dari dataset insurance yang berisi 7 kolom dengan detail: `age`, `sex`, `bmi`, `children`, `smoker`, `region`, dan `charges` berikut visualisasi yang dapat ditampilkan dengan berbagai *trace* pada *Plotly Graph Objects*.

1. *Pie Chart* (menampilkan proporsi dari total dalam bentuk irisan).
   Dalam dataset "insurance" dapat menggunakan *pie chart* untuk melihat distribusi usia, di mana sebelumnya dilakukan *binning* terlebih dahulu. *Pie chart* yang dihasilkan interaktif, apabila kursor diarahkan pada area *pie chart* dimaksud, akan melakukan *hover* informasi, berapa jumlah distribusi pada usia tertentu dan berapa presentase distribusinya dari keseluruhan.

```
# 1. Distribusi Usia (Pie Chart)
age_bins = [0, 18, 30, 40, 50, 60, 100]
age_labels = ['0-18', '19-30', '31-40', '41-50', '51-60', '61+']
df['age_group'] = pd.cut(df['age'], bins=age_bins, labels=age_labels, right=False)
age_distribution = df['age_group'].value_counts()
fig = go.Figure(data=[go.Pie(labels=age_distribution.index, values=age_distribution.values)])
fig.update_layout(title='Distribusi Usia')
fig.show()
```
✓ 0.3s

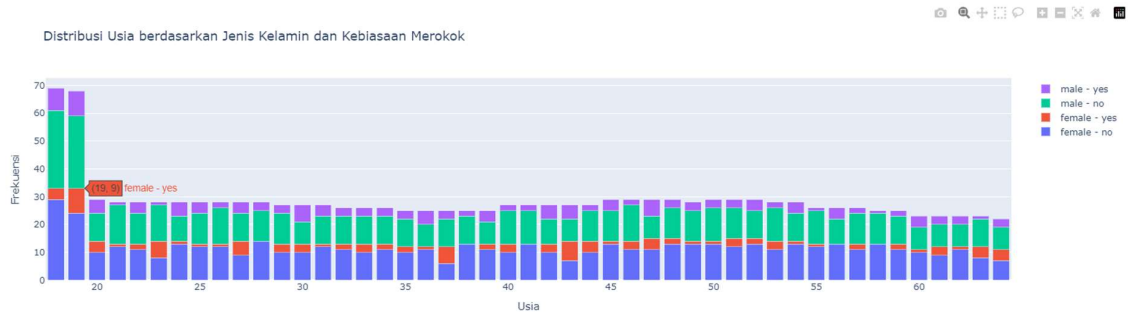


2. *Stacked Bar Chart* (menampilkan data dalam bentuk batang vertikal atau horizontal yang bersusun).
   Contoh *stacked bar chart* pada dataset *insurance*, melihat distribusi usia dan jenis kelamin berdasarkan kebiasaan merokok.

```python
# 2. Distribusi Usia dan Jenis Kelamin berdasarkan kebiasaan merokok (stacked bar chart)
# Menghitung frekuensi untuk stacked bar chart
df_grouped = df.groupby(['age', 'sex', 'smoker']).size().reset_index(name='count')
fig2 = go.Figure()

# Menambahkan trace untuk setiap kategori Sex dan Smoker
for sex in df_grouped['sex'].unique():
    df_sex = df_grouped[df_grouped['sex'] == sex]
    for smoker_status in df_sex['smoker'].unique():
        df_smoker = df_sex[df_sex['smoker'] == smoker_status]
        fig2.add_trace(go.Bar(
            x=df_smoker['age'],
            y=df_smoker['count'],
            name=f'{sex} - {smoker_status}',
            hoverinfo='x+y+name'
        ))
fig2.update_layout(
    title='Distribusi Usia berdasarkan Jenis Kelamin dan Kebiasaan Merokok',
    xaxis_title='Usia',
    yaxis_title='Frekuensi',
    barmode='stack')
fig2.show()
```

✓ 0.0s

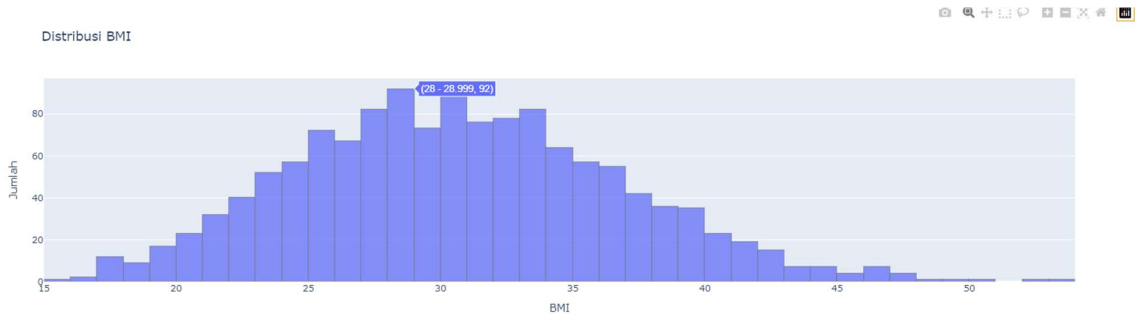


Hasil yang ditampilkan pada stacked bar chart dapat memuat informasi seberapa banyak distribusi `sex` laki-laki atau peremuan dengan kebiasaan merokok di masing-masing usia, sehingga terlihat proporsi seberapa banyak komposisinya di masing-masing usia.

3. Histogram (menggambarkan distribusi frekuensi data.)
   Dalam contoh dataset *insurance*, histogram dapat digunakan untuk memvisualisasikan data distribusi BMI.

```python
# 3.Distribusi BMI (Histogram)
fig3 = go.Figure()
fig3.add_trace(go.Histogram(
    x=df['bmi'],
    name='BMI',
    opacity=0.75,
    marker=dict(line=dict(width=0.5)),  # Garis tepi pada bar
))
fig3.update_layout(
    title='Distribusi BMI',
    xaxis_title='BMI',
    yaxis_title='Jumlah',
)
fig3.show()
```

✓ 0.0s

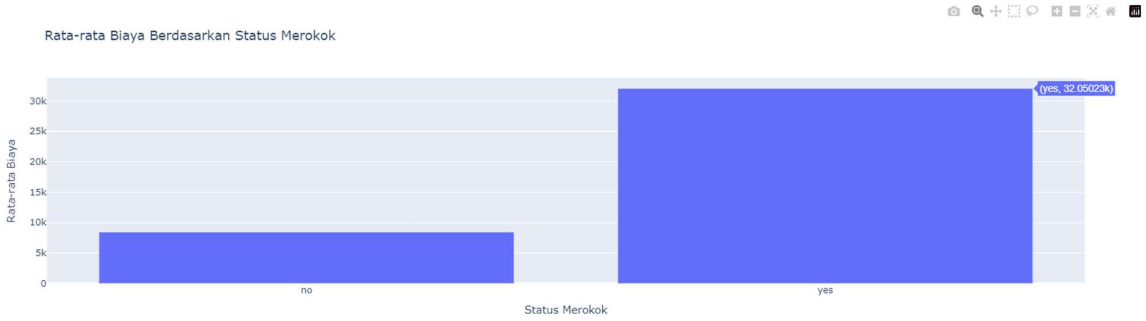


Hasil visualisasi histogram dari Distribusi BMI dapat melakukan *hover* informasi berapa jumlah orang pada BMI tertentu, misalnya pada BMI 28 terdapat 92 orang.

4. *Bar Chart* (menampilkan data dalam bentuk batang vertikal atau horizontal).
   Contohnya apabila ingin menampilkan berapa rata-rata biaya atau `charges` berdasarkan status merokok.

```
# 4. Rata-rata biaya berdasarkan status merokok (Bar Chart)
avg_charges_smoker = df.groupby('smoker')['charges'].mean().reset_index()
fig4 = go.Figure()
fig4.add_trace(go.Bar(x=avg_charges_smoker['smoker'], y=avg_charges_smoker['charges']))
fig4.update_layout(title='Rata-rata Biaya Berdasarkan Status Merokok', xaxis_title='Status Merokok', yaxis_title='Rata-rata Biaya')
fig4.show()
```

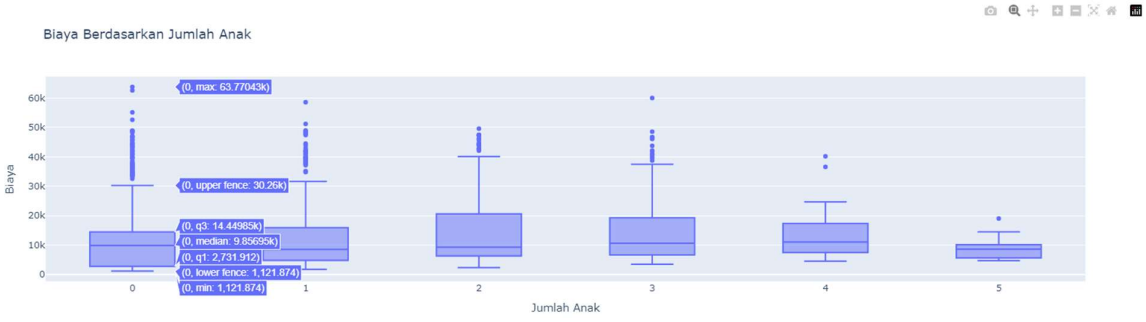
✓ 0.0s



5. *Box Plot* (menampilkan ringkasan statistik data, seperti median dan kuartil).
   Pada dataset, dapat dilihat besaran masing-masing biaya berdasarkan jumlah anak.

```
# 5. Biaya berdasarkan jumlah anak
fig5 = go.Figure()
fig5.add_trace(go.Box(x=df['children'], y=df['charges']))
fig5.update_layout(title='Biaya Berdasarkan Jumlah Anak', xaxis_title='Jumlah Anak', yaxis_title='Biaya')
fig5.show()
```

✓ 0.0s



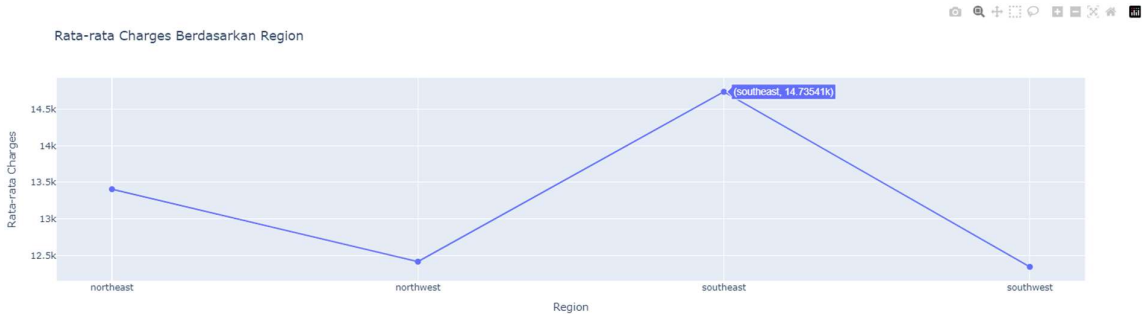
Dalam visualisasi *box plot* ini ditampilkan *hover* informasi statistik misalnya pada data jumlah anak tertentu, berapa masing-masing nilai biaya / charges min, max, median, kuartil 1, kuartil 3, *outlier* bahkan *upper fence* nya.

6. *Line Plot* (menghubungkan titik data dengan garis).
   Line plot pada dataset insurance dapat memvisualisasikan rata-rata biaya per regionya.

```
# 6. Rata-rata biaya (charges) per region (Line chart)
average_charges = df.groupby('region')['charges'].mean().reset_index()
fig6 = go.Figure()
fig6.add_trace(go.Scatter(
    x=average_charges['region'],
    y=average_charges['charges'],
    mode='lines+markers',
    name='Rata-rata Charges',
    line=dict(shape='linear'),
    marker=dict(size=8)
))
fig6.update_layout(
    title='Rata-rata Charges Berdasarkan Region',
    xaxis_title='Region',
    yaxis_title='Rata-rata Charges',
)
fig6.show()
```

✓ 0.0s

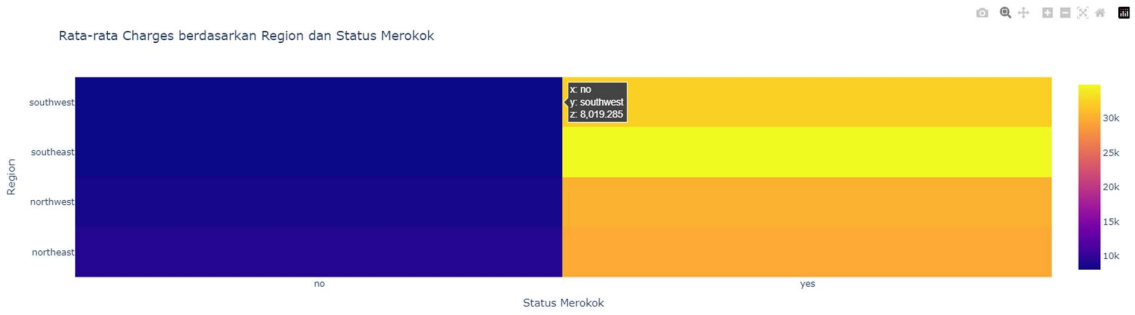
Pada visualisasi line plot menggunakan *Graph Objects* ini dapat mengetahui masing-masing region berikut besaran nilai rata-rata biaya / `charges`.



7. *Heatmap* (**m**enunjukkan nilai data dalam format matriks dengan warna).
   Contoh heatmap dari dataset *insurance* yaitu rata-rata biaya untuk region dan status merokok.

```python
# 7. Rata-rata charges untuk setiap kombinasi region dan smoker
heatmap_data = df.groupby(['region', 'smoker'])['charges'].mean().unstack()
fig7 = go.Figure(data=go.Heatmap(
    z=heatmap_data.values,
    x=heatmap_data.columns,
    y=heatmap_data.index,
    colorscale='Plasma'
))
fig7.update_layout(
    title='Rata-rata Charges berdasarkan Region dan Status Merokok',
    xaxis_title='Status Merokok',
    yaxis_title='Region',
)
fig7.show()
```

✓ 0.0s



Hover informasi pada heatmap di dalam contoh, dapat menampilkan rata-rata charges pada masing-masing kategori region dan status merokok.

8. *Scatter Plot* (memvisualisasikan data dalam bentuk titik-titik).
   Scatter plot pada dataset *insurance* dapat memvisualisasikan usia berbanding dengan biaya / `charges`.

```python
# 8. Usia Vs Biaya (scatter plot)
fig8 = go.Figure()
fig8.add_trace(go.Scatter(
    x=df['age'],
    y=df['charges'],
    mode='markers',
    marker=dict(
        size=10,
        color=df['bmi'],  # Warna berdasarkan BMI
        colorscale='Plasma',  # Skala warna
        colorbar=dict(title='BMI'),  # Menambahkan color bar
        showscale=True
    ),
))
fig8.update_layout(
    title='Scatter Plot: Usia vs Biaya',
    xaxis_title='Usia',
    yaxis_title='Biaya',
)
fig8.show()
```

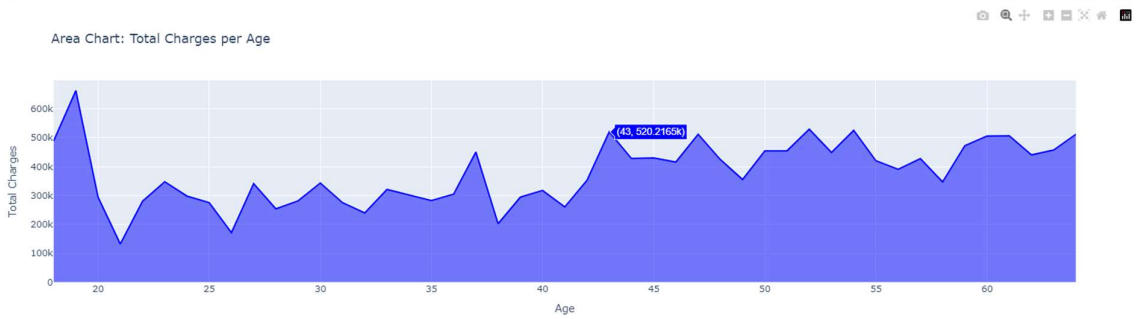
✓ 0.0s

Hover informasi pada *scatter plot* dapat berisi berapa biaya untuk masing-masing usia secara detail.

9. *Area chart* (mirip dengan *line plot*, tetapi area di bawah garis diwarnai / diarsir warna).
   Pada dasarnya untuk *area chart* tetap menggunakan traces `go.Scatter` namun menambahkan `mode`, `fill` dan `line` sehingga menampilkan visualisasi *area chart*. Dari dataset insurance, dapat menampilkan total biaya per usia menggunakan *area chart*.

```python
# 9. Total Biaya per Usia (Area Chart)
total_charges = df.groupby('age')['charges'].sum().reset_index()
fig9 = go.Figure()
fig9.add_trace(go.Scatter(
    x=total_charges['age'],
    y=total_charges['charges'],
    mode='lines',
    fill='tozeroy',  # Mengisi area di bawah garis
    name='Total Charges',
    line=dict(color='Blue'),
    marker=dict(size=8)
))
fig9.update_layout(
    title='Area Chart: Total Charges per Age',
    xaxis_title='Age',
    yaxis_title='Total Charges',
)
fig9.show()
```

✓ 0.0s

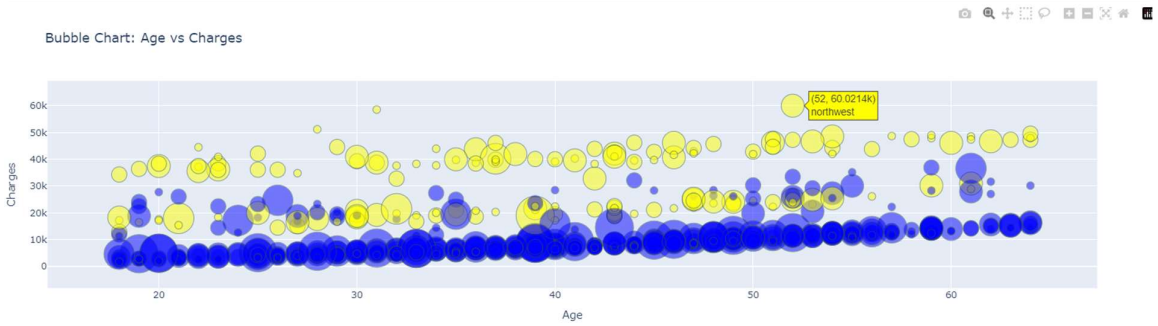


Hover informasi pada visualisasi *area chart* Total biaya / charges per age, berisi berapa biaya untuk usia tertentu.

10. *Bubble Chart* (*scatter plot* dengan ukuran titik yang mewakili nilai tambahan).
    Mirip dengan *scatter plot*, *bubble chart* juga menggunakan *traces* `go.Scatter` menambahkan `size` untuk mengatur ukuran *bubble*.

```python
# 10. Usia vs Biaya berdasarkan Region dan Status Merokok (Bubble Chart)
fig10 = go.Figure()
fig10.add_trace(go.Scatter(
    x=df['age'],
    y=df['charges'],
    mode='markers',
    marker=dict(
        size=df['children'] * 10,  # Ukuran bubble berdasarkan jumlah anak, dikalikan untuk memperbesar tampilan
        color=df['smoker'].map({'yes': 'yellow', 'no': 'blue'}),  # Warna berdasarkan status merokok
        opacity=0.5,
        line=dict(width=1, color='DarkSlateGrey')
    ),
    text=df['region'],  # Menampilkan region saat hover
    name='Data Points'
))
fig10.update_layout(
    title='Bubble Chart: Age vs Charges',
    xaxis_title='Age',
    yaxis_title='Charges',
    showlegend=False  # Menyembunyikan legend karena sudah ada informasi di hover
)
fig10.show()
```

✓ 0.1s



Bubble chart pada data insurance memuat informasi usia berbanding biaya (*Age vs Charges*) per region. *Hover* informasi pada visualisasi menampilkan detail berapa biaya untuk kategori masing-masing region dan usia.

11. Surface Plot (visualisasi data tiga dimensi dengan permukaan halus.)
    Visualisai ini termasuk dalam visualisasi 3D yang mencakup misalnya pada dataset *insurance* adalah biaya, usia dan bmi (biaya berdasarkan usia dan BMI). Traces yang digunakan pada Plotly Graph Objects ini adalah `go.Surface`.

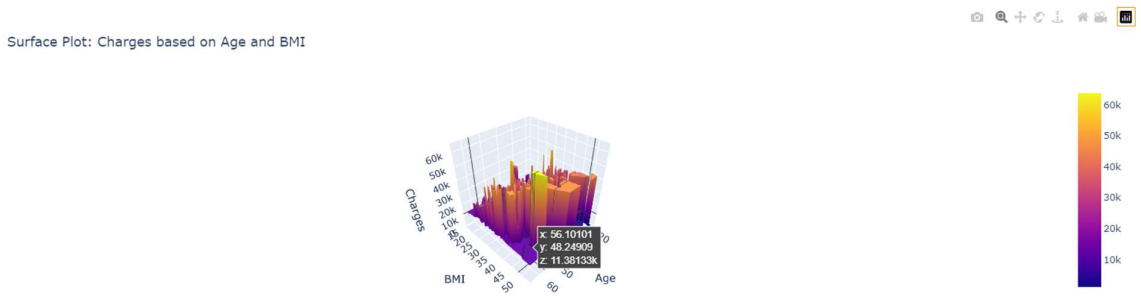
```python
# 11. Biaya berdasarkan Usia dan BMI (Surface Plot)
# Membuat grid untuk surface plot
age_range = np.linspace(df['age'].min(), df['age'].max(), 100)
bmi_range = np.linspace(df['bmi'].min(), df['bmi'].max(), 100)
age_grid, bmi_grid = np.meshgrid(age_range, bmi_range)

# Menghitung charges berdasarkan interpolasi (andaikan linear)
charges_grid = np.zeros_like(age_grid)

for i in range(len(age_range)):
    for j in range(len(bmi_range)):
        # Menggunakan nilai terdekat untuk interpolasi
        closest = ((df['age'] - age_range[i])**2 + (df['bmi'] - bmi_range[j])**2).idxmin()
        charges_grid[j, i] = df['charges'].iloc[closest]

# Membuat surface plot
fig11 = go.Figure(data=[go.Surface(z=charges_grid, x=age_grid, y=bmi_grid)])
fig11.update_layout(
    title='Surface Plot: Charges based on Age and BMI',
    scene=dict(
        xaxis_title='Age',
        yaxis_title='BMI',
        zaxis_title='Charges'
    )
)
fig11.show()
```

✓ 3.2s

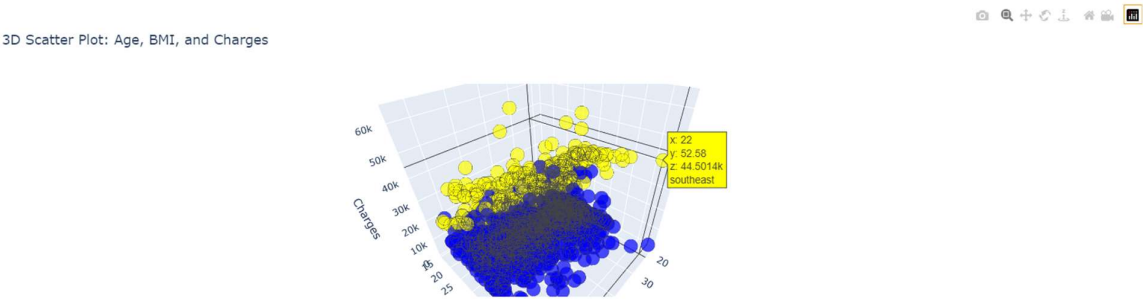


Visualisasi pada *surface plot* biaya berdasarkan usia dan BMI, dapat menampilkan hover informasi x (usia), y (BMI), z (biaya). Akan tampil juga secara 3D biaya yang tertinggi diwakilkan dengan warna tertentu.

12. *3D Scatter Plot* (*scatter plot* dalam tiga dimensi).
Traces pada *3D scatter plot* menggunakan `go.Scatter3D` dengan mendefinisikan nilai-nilai `x, y` dan `z` –nya. Pada dataset *insurance*, visualisasi menggunakan *3D scatter plot* dapat menampilkan usia, BMI dan biaya pada masing-masing region.

```
# 12. 3D Scatter Plot: Age, BMI, and Charges
fig12 = go.Figure()
fig12.add_trace(go.Scatter3d(
    x=df['age'],
    y=df['bmi'],
    z=df['charges'],
    mode='markers',
    marker=dict(
        size=10,
        color=df['smoker'].map({'yes': 'yellow', 'no': 'blue'}),  # Warna berdasarkan status merokok
        opacity=0.7,
        line=dict(width=1)
    ),
    text=df['region'],  # Menampilkan region saat hover
))
fig12.update_layout(
    title='3D Scatter Plot: Age, BMI, and Charges',
    scene=dict(
        xaxis_title='Age',
        yaxis_title='BMI',
        zaxis_title='Charges'
    )
)
fig12.show()
```

✓ 0.1s



Setiap visualisasi 3D dapat melakukan zoom in lebih detail dan interaktif, berbeda dengan visualisasi 2D. Hover informasinya juga dapat dikustomisasi sesuai kebutuhan. Pada contoh ini adalah usia, BMI, dan biaya di masing-masing regionnya.

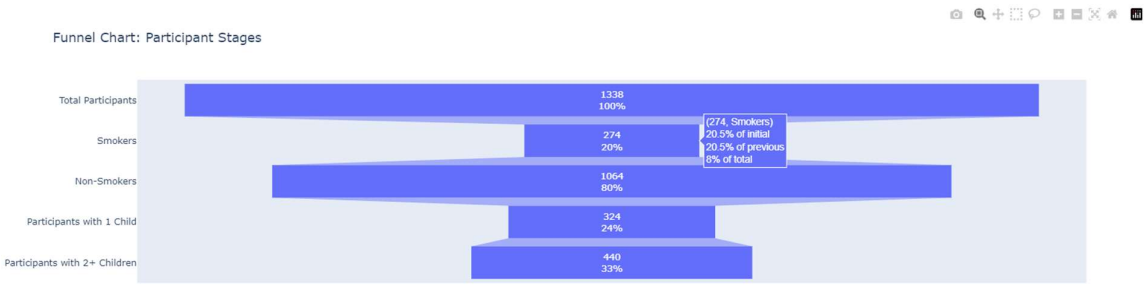
13. *Funnel Chart* (menunjukkan alur proses atau konversi).
    Contoh funnel chart pada dataset insurance misalnya, *participants stages* berdasarkan kategori status merokok dan jumlah anak. *Traces* yang digunakan adalah `go.Funel`.

```
# 13. Stages Participants, total peserta berdasarkan kategori Status Merokok dan Jumlah Anak (Funnel Chart)
# Menghitung jumlah berdasarkan kategori
total_count = len(df)
smoker_count = df['smoker'].value_counts().get('yes', 0)
non_smoker_count = df['smoker'].value_counts().get('no', 0)

# Menghitung total berdasarkan jumlah anak
children_counts = df['children'].value_counts().sort_index()

# Menyiapkan data untuk funnel chart
stages = ['Total Participants', 'Smokers', 'Non-Smokers', 'Participants with 1 Child', 'Participants with 2+ Children']
values = [total_count, smoker_count, non_smoker_count, children_counts.get(1, 0), children_counts[2:].sum()]

fig13 = go.Figure()
fig13.add_trace(go.Funnel(
    name='Funnel Chart',
    y=stages,
    x=values,
    textinfo='value+percent initial',  # Menampilkan nilai dan persentase
))
fig13.update_layout(
    title='Funnel Chart: Participant Stages',
)
fig13.show()
✓ 0.0s
```



Visualisasi *funnel chart* biasanya digunakan untuk melihat *stages* / tahapan/ tingkatan tertentu. Pada contoh di atas, *funnel chart* membantu melihat komposisi berapa banyak partisipan dari status merokok dan tidak merokok, juga berapa banyak yang sudah memiliki 1 anak atau memiliki anak lebih dari 2.

Kesimpulan dan Penutup

Plotly Graph Objects adalah salah satu pustaka dalam bahasa *Python* untuk menciptakan visualisasi data yang interaktif dan menarik. Dengan berbagai fitur dan fungsionalitasnya, pustaka ini mendukung eksplorasi data, visualisasi ilmiah, dan banyak aplikasi lainnya, sehingga memungkinkan pengguna untuk menganalisis dan menyampaikan wawasan dari data yang ada secara efektif. Pengguna dapat menggunakan modul ini jika perlu membuat plot yang cukup rumit dan memerlukan kustomisasi yang mendetail. Selain itu, pengguna juga dapat memanfaatkan fitur-fitur lanjutnya seperti jejak, bentuk khusus, anotasi, atau tata letak lain yang cukup rumit.

Referensi

AIMYSTERIES. (2022). *What are The Graph Objects in Plotly and How to use them ?.* Diakses pada 31 Oktober 2024, dari https://analyticsindiamag.com/ai-mysteries/what-are-the-graph-objects-in-plotly-and-how-to-use-them/

Educative. *Plotly Graph Objects and its methods.* Diakses pada 31 Oktober 2024, dari https://www.educative.io/answers/plotly-graph-objects-and-its-methods

Geeksforgeeks. (2023). *Plotly Tutorial*. Diakses pada 31 Oktober 2024, dari https://www.geeksforgeeks.org/python-plotly-tutorial/

Geeksforgeeks. (2024). *Getting Started With Plotly Python*. Diakses pada 31 Oktober 2024, dari https://www.geeksforgeeks.org/getting-started-with-plotly-python/

Holtz, Yan. *Interactive Charts with Plotly.* Diakses pada 31 Oktober 2024, dari https://python-graph-gallery.com/plotly/

Plotly Graphing Libraries. *Graph Objects in Python.* Diakses pada 31 Oktober 2024, dari https://plotly.com/python/graph-objects/

Pypi.org. *Plotly 5.24.1*. Diakses pada 31 Oktober 2024, dari https://pypi.org/project/plotly/

Research Bazaar (ResBaz). *Our First Plotly Graphs.* Diakses pada 31 Oktober 2024, dari https://maegul.gitbooks.io/resguides-plotly/content/content/plotting_locally_and_offline/python/customising_our_plotly_graphs.html

Tutorialspoint. *Plotly – Package Structure.* Diakses pada 31 Oktober 2024, dari https://www.tutorialspoint.com/plotly/plotly_package_structure.htm