Task-3 Data Visualization Dashboard

Task Description: 1.data visualiztion dashboard using a tool Plotly Dash. 2.EDA and showcase the results of predictive modeling task. 3.dashboard to be user-friendly and informative.

1.data visualiztion dashboard using a tool Plotly Dash.

In [ ]:
```
pip install dash
```

In [ ]:
```
import pandas as pd
```

In [ ]:
```
# Load the Titanic dataset
titanic_df = pd.read_csv('titanic.csv')
titanic_df
```

In [ ]:
```
import dash
from dash import dcc, html
from dash.dependencies import Input, Output
import plotly.express as px
import pandas as pd

# Load your dataset (titanic_df) and perform EDA

# Create a Dash app
app = dash.Dash(__name__)

# Define layout
app.layout = html.Div([
    html.H1("Interactive Data Visualization Dashboard"),

    # Dropdown menu to select feature for x-axis
    dcc.Dropdown(
        id='x-axis-dropdown',
        options=[
            {'label': col, 'value': col} for col in titanic_df.columns
        ],
        value='Age',  # Default value
        clearable=False,
    ),

    # Scatter plot
    dcc.Graph(id='scatter-plot'),
])

# Define callback to update scatter plot
@app.callback(
    Output('scatter-plot', 'figure'),
    [Input('x-axis-dropdown', 'value')]
)
def update_scatter_plot(selected_feature):
    # Create scatter plot
    fig = px.scatter(
        data_frame=titanic_df,
        x=selected_feature,
        y='Survived',  # Assuming 'Survived' is the target variable in your dataset
        title=f"Scatter Plot of {selected_feature} vs. Survived",
        labels={'Survived': 'Survival Status'},
        hover_data=[selected_feature, 'Survived'],  # Additional info on hover
    )
    return fig

# Run the app
if __name__ == '__main__':
    app.run_server(debug=True)
```

In [ ]:

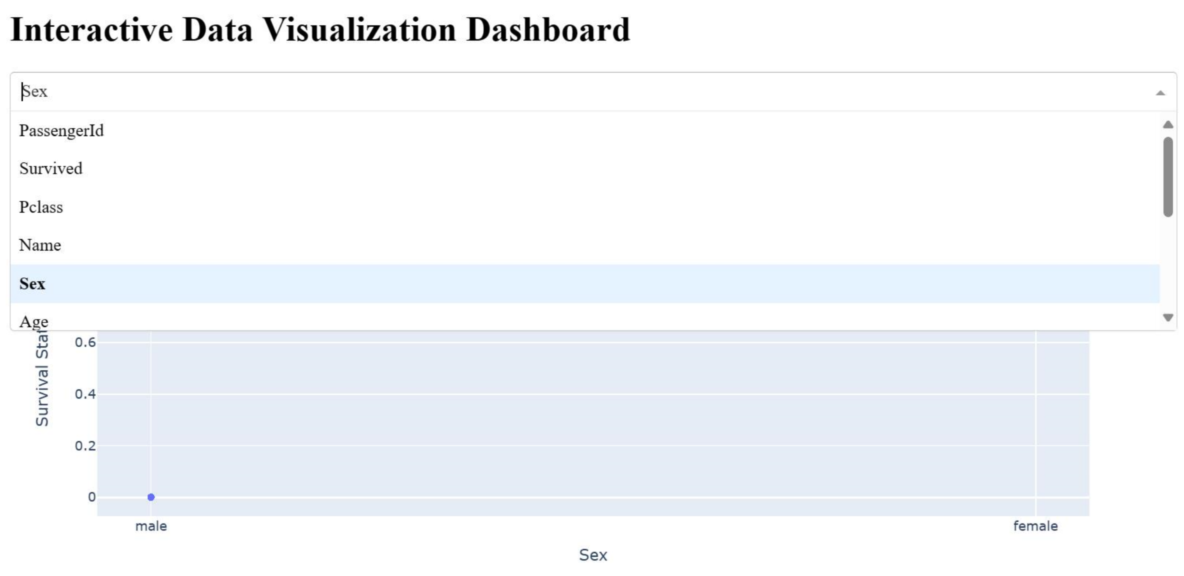
2.EDA and showcase the results of predictive modeling task.

In [ ]:
```
import dash
from dash import dcc, html
from dash.dependencies import Input, Output
import plotly.express as px
import pandas as pd

# Load your dataset (titanic_df) and perform EDA

# Create a Dash app
app = dash.Dash(__name__)

# Define layout
app.layout = html.Div([
    html.H1("Interactive Data Visualization Dashboard"),

    # Dropdown menu to select plot type
    dcc.Dropdown(
        id='plot-type-dropdown',
        options=[
            {'label': 'Bar Chart', 'value': 'bar_chart'},
            {'label': 'Line Chart', 'value': 'line_chart'},
            {'label': 'Heatmap', 'value': 'heatmap'},
            {'label': 'Box Plot', 'value': 'box_plot'},
        ],
        value='bar_chart',  # Default value
        clearable=False,
    ),

    # Output container for plot
    html.Div(id='plot-container'),
])

# Define callback to update plot based on plot type
@app.callback(
    Output('plot-container', 'children'),
    [Input('plot-type-dropdown', 'value')]
)
def update_plot(plot_type):
    if plot_type == 'bar_chart':
        # Bar Chart
        fig = px.bar(titanic_df, x='Pclass', y='Fare', title='Bar Chart of Fare by Pclass', color='Pclass')

    elif plot_type == 'line_chart':
        # Line Chart
        line_df = titanic_df.groupby('Age')['Fare'].mean().reset_index()
        fig = px.line(line_df, x='Age', y='Fare', title='Line Chart of Age vs. Mean Fare', color_discrete_sequence=px.colors.qualitative.Dark2)

    elif plot_type == 'heatmap':
        # Heatmap
        fig = px.imshow(titanic_df.corr(), title='Heatmap of Correlation Matrix', color_continuous_scale=px.colors.diverging.RdBu)

    elif plot_type == 'box_plot':
        # Box Plot
        fig = px.box(titanic_df, x='Pclass', y='Age', title='Box Plot of Age by Pclass', color='Pclass')

    else:
        fig = None

    if fig is not None:
        return dcc.Graph(figure=fig)
    else:
        return html.P("Select a plot type from the dropdown to display.")

# Run the app
if __name__ == '__main__':
    app.run_server(debug=True)
```

In [ ]:

3.dashboard to be user-friendly and informative.

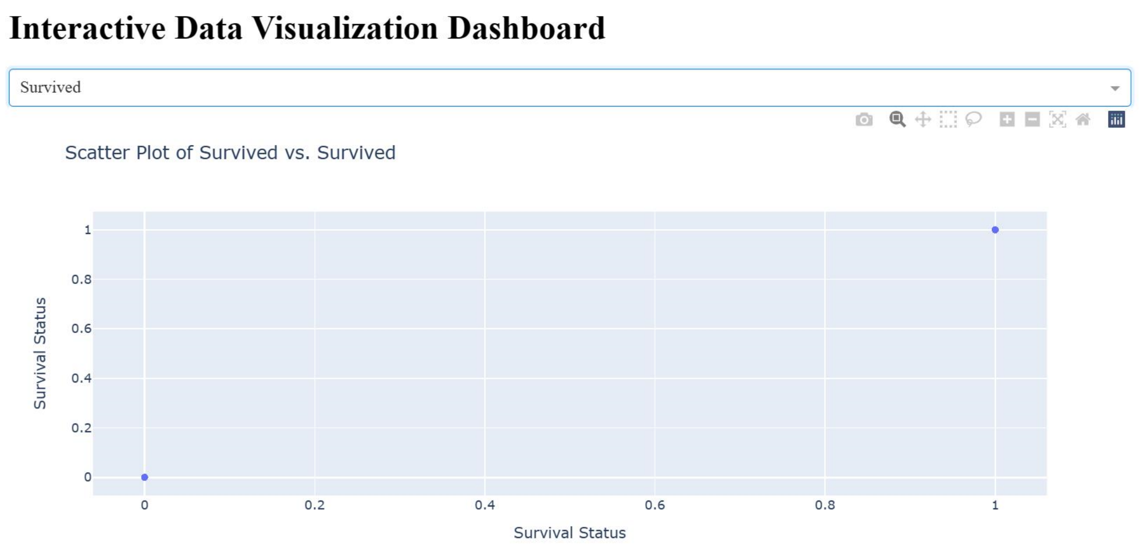
In [ ]:
```
import dash
from dash import dcc, html
from dash.dependencies import Input, Output
import plotly.express as px
import pandas as pd

# Load your dataset (titanic_df) and perform EDA

# Create a Dash app
app = dash.Dash(__name__)

# Define layout
app.layout = html.Div([
    html.H1("Interactive Data Visualization Dashboard"),

    # Dropdown menu to select plot type
    dcc.Dropdown(
        id='plot-type-dropdown',
        options=[
            {'label': 'Violin Plot', 'value': 'violin_plot'},
            {'label': 'Area Chart', 'value': 'area_chart'},
        ],
        value='violin_plot',  # Default value
        clearable=False,
        style={'width': '50%'}  # Adjust the width of the dropdown
    ),

    # Output container for plot
    html.Div(id='plot-container'),
], style={'textAlign': 'center', 'margin': '50px auto'})  # Center the layout and add margin

# Define callback to update plot based on plot type
@app.callback(
    Output('plot-container', 'children'),
    [Input('plot-type-dropdown', 'value')]
)
def update_plot(plot_type):
    if plot_type == 'violin_plot':
        # Violin Plot
        fig = px.violin(titanic_df, y='Age', x='Pclass', title='Violin Plot of Age by Pclass',
                        color='Pclass', box=True, points='all', hover_data=['Fare'],
                        labels={'Age': 'Age (Years)'})

    elif plot_type == 'area_chart':
        # Area Chart
        area_df = titanic_df.groupby('Age')['Fare'].sum().reset_index()
        fig = px.area(area_df, x='Age', y='Fare', title='Area Chart of Fare by Age',
                      color_discrete_sequence=px.colors.qualitative.Pastel,
                      labels={'Fare': 'Total Fare (USD)'})

    else:
        fig = None

    if fig is not None:
        fig.update_layout(  # Update layout for all plots
            plot_bgcolor='rgba(0, 0, 0, 0)',  # Set plot background color to transparent
            paper_bgcolor='#f8f9fa',  # Set paper background color
            font=dict(family="Arial, sans-serif", size=12, color="#505050"),  # Set font style and size
            margin=dict(t=50, r=50, b=50, l=50),  # Set margin for the plot
            hoverlabel=dict(bgcolor="white", font_size=12, font_family="Arial"),  # Set hover label style
        )
        return dcc.Graph(figure=fig)
    else:
        return html.P("Select a plot type from the dropdown to display.")

# Run the app
if __name__ == '__main__':
```



Interactive Data Visualization Dashboard

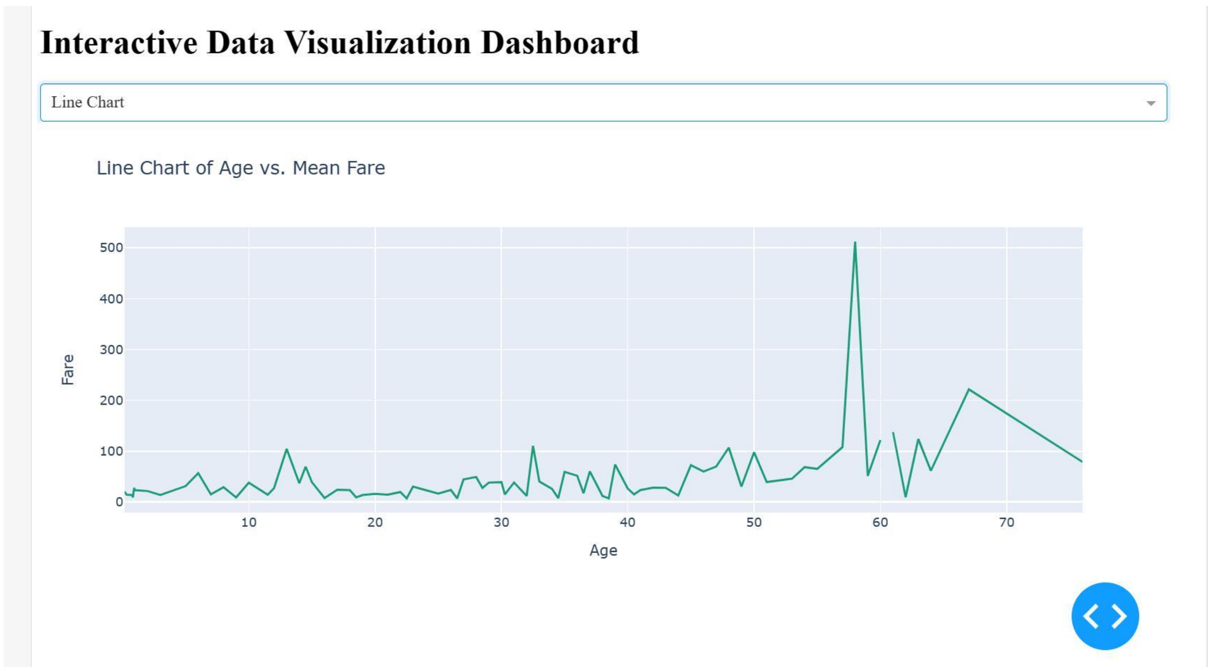
Sex

- PassengerId
- Survived
- Pclass
- Name
- **Sex**
- Age



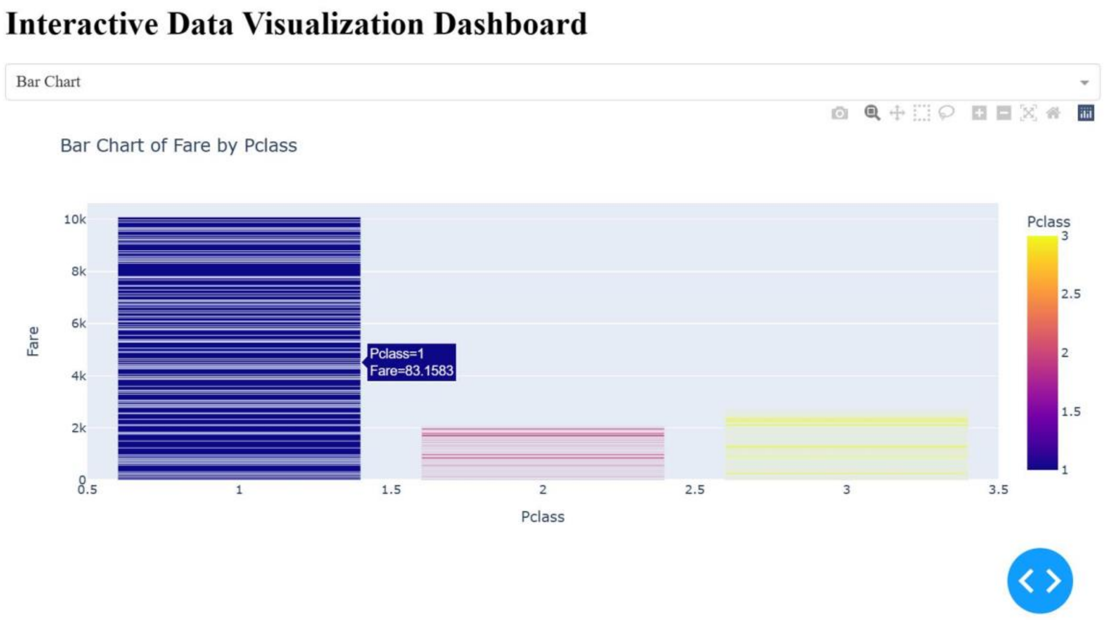
Interactive Data Visualization Dashboard

Survived



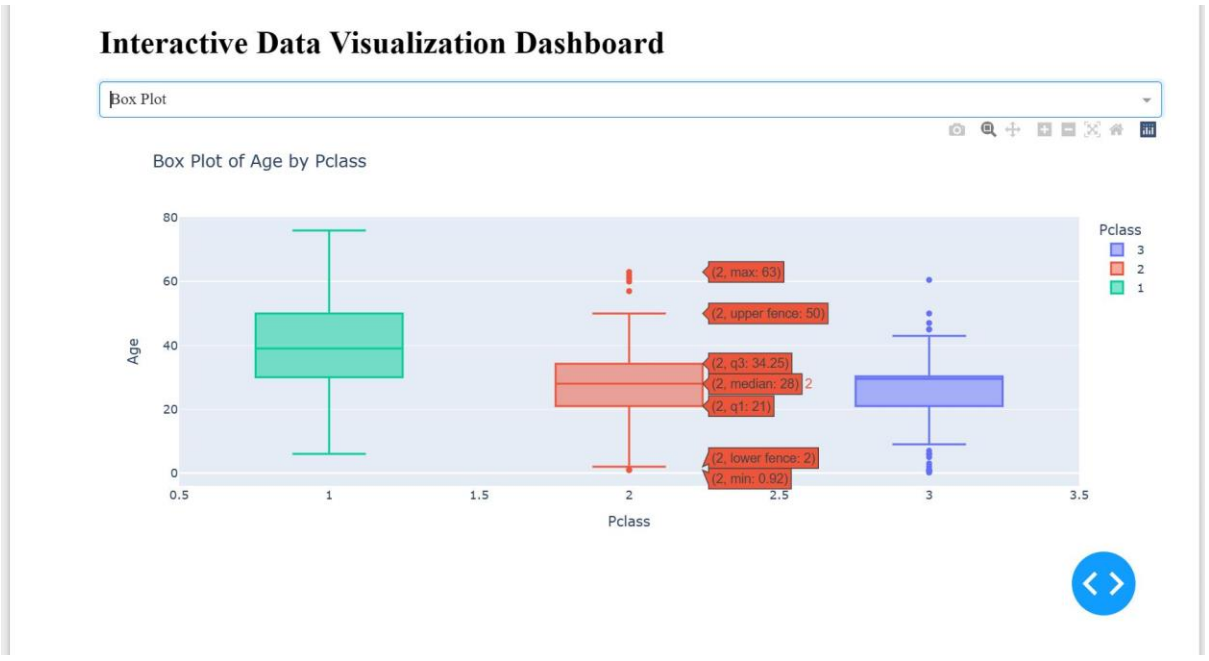
Interactive Data Visualization Dashboard

Line Chart

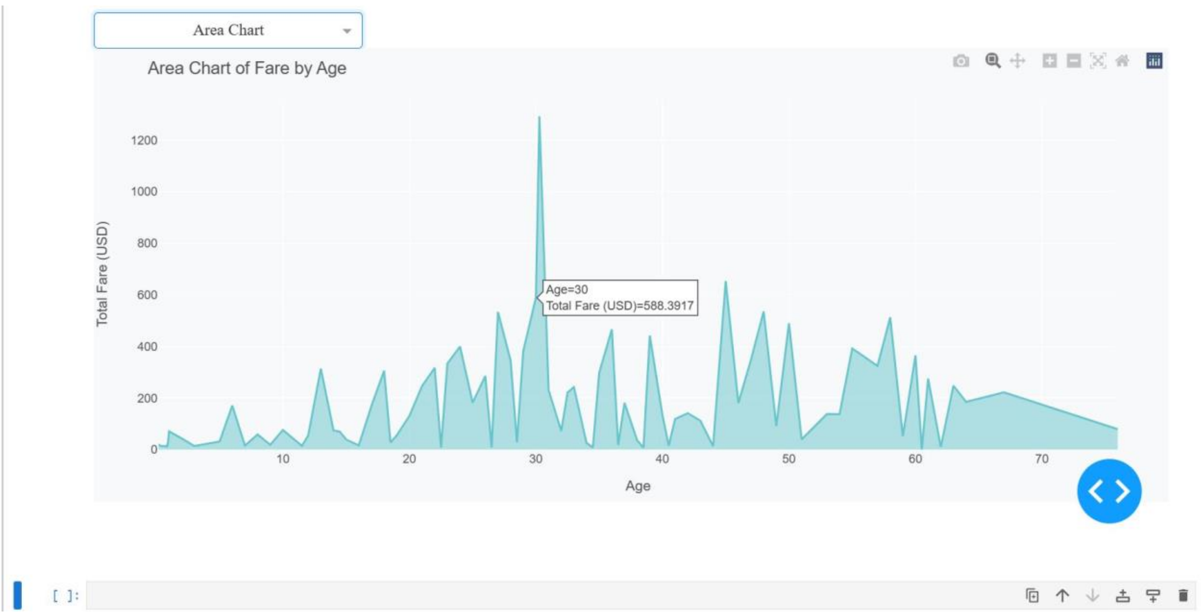


Interactive Data Visualization Dashboard

Bar Chart



Interactive Data Visualization Dashboard



Area Chart
Area Chart of Fare by Age
Total Fare (USD)
Age
Age=30
Total Fare (USD)=588.3917
1200
1000
800
600
400
200
0
10
20
30
40
50
60
70
[ ]: