CSCI6612 - Visual Analytics

Assignment 2

Fall 2023

Due on: 6 October 2023, 23:59 ADT

Name: Abhinav Acharya Tirumala Vinjamuri
Banner ID: B00929073
Email: ab806657@dal.ca

Name: Arihant Dugar
Banner ID: B00917961
Email: ar968345@dal.ca

QUESTION 1

In this question, you study the Dash tutorial, and you will experiment with some basic visualization algorithms that you will embed within Dash. You will continue using the breast cancer dataset. Make use of code examples available in the Dash tutorial and Plotly documentation, and clearly reference the URLs of the classes / methods you reused.

Before you start this assignment, study the Dash in 20 minutes tutorial and replicate the examples in a notebook. Skip the examples that require a Dash Enterprise license. Look up the documentation of the methods and libraries used. A particularly useful library is Plotly Express, which supports the most common plot types, with interactive controls. The Plotly Express cheat sheet is useful. Aman has put together a resource document on Notion that he will keep updating as the course advances.

In [1]:

```
import sklearn
import pandas as pd
import numpy as np
from sklearn.datasets import load_breast_cancer
from plotly.subplots import make_subplots
import plotly.graph_objects as go
import dash
from dash import dcc
from dash import html
import plotly.express as px
from dash.dependencies import Input, Output
import umap.umap_ as umap
```

1. Load the breast cancer dataset, and convert the loaded data to a pandas dataframe, as it plays well with Plotly Express.

In [2]:

```
#Load breast cancer dataset
data = load_breast_cancer()
df = pd.DataFrame(np.c_[data['data'], data['target']], columns= np.append(data['feature_names'], ['target']))

display(df.head())

display(df.describe())
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	wo perime
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	17.33	184.
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	23.41	158.
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	25.53	152.
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	26.50	98.
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	16.67	152.

5 rows × 31 columns

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fracta dimensior
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.00000
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181162	0.06279
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414	0.00706
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	0.04996
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	0.05770
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	0.06154
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700	0.06612
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	0.09744

8 rows × 31 columns

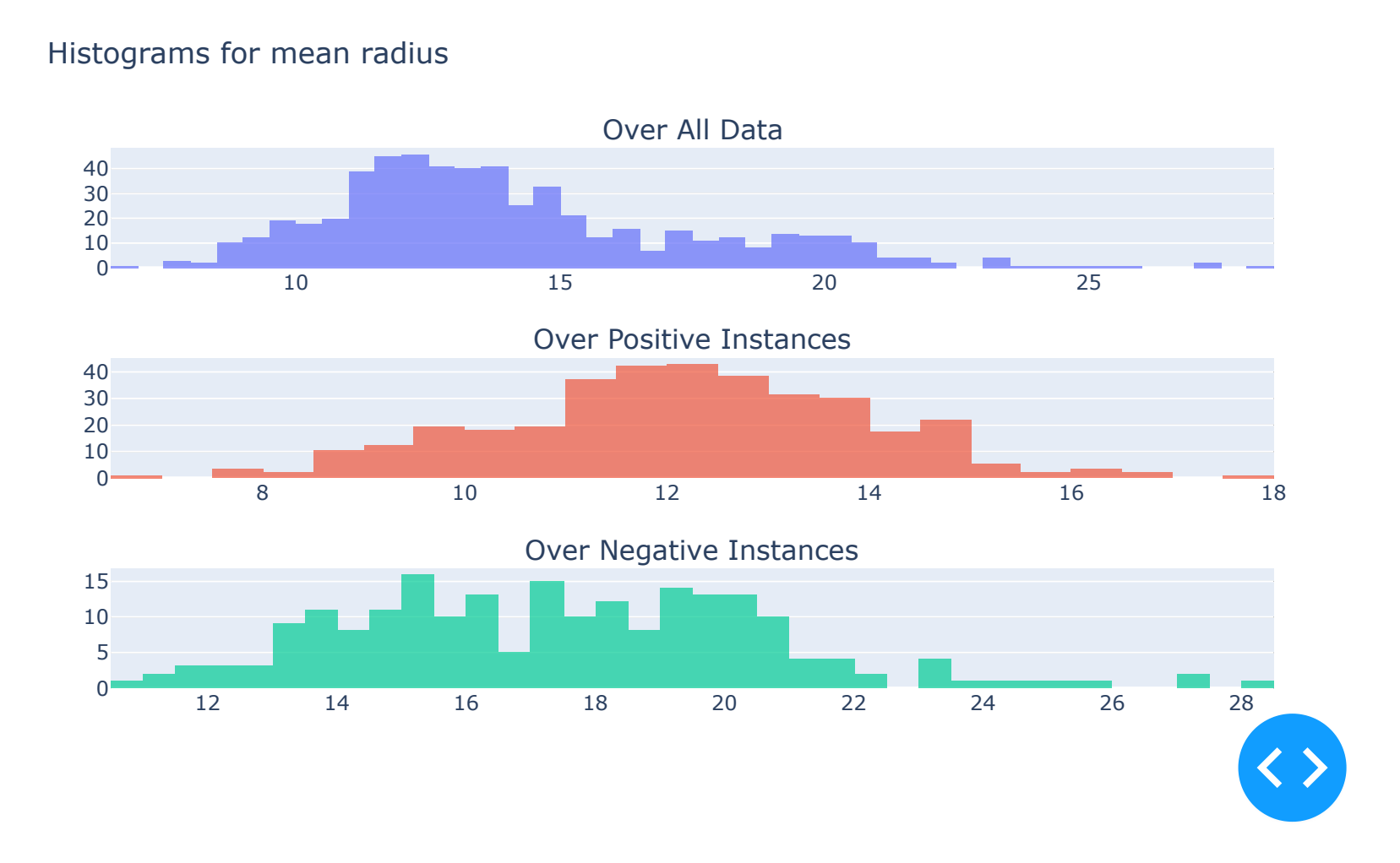
2. Using Dash, create an interactive tool that allows the user to select a single feature among the set of features and plot three histograms, first, over all the data, second, over the positive instances, third, over the negative instances. Arrange the three histograms vertically, with the same bins, so that they are easier to visually compare. Using your tool, inspect the histograms of all the features, and describe your observations, in the context of the classification task. Hint: You may find it helpful to use subplots, https://plotly.com/python/subplots/

In [3]:

```
app = dash.Dash(__name__)

app.layout = html.Div([
    html.H1("Featured Histograms for Breast Cancer Dataset"),
    dcc.Dropdown(
        id='feature-selector',
        options=[{'label': col, 'value': col} for col in df.columns[:-1]],
        value=df.columns[0],
    ),

    dcc.Graph(id='histogram-subplots')
])

@app.callback(
    Output('histogram-subplots', 'figure'),
    Input('feature-selector', 'value')
)
def update_histogram(selected_feature):
    fig = make_subplots(rows=3, cols=1, subplot_titles=("Over All Data", "Over Positive Instances", "Over Negative Inst

    all_data_hist = go.Histogram(x=df[selected_feature], opacity=0.7, nbinsx=50, name="All Data")
    positive_data_hist = go.Histogram(x=df[df['target'] == 1][selected_feature], opacity=0.7, nbinsx=50, name="Positive
    negative_data_hist = go.Histogram(x=df[df['target'] == 0][selected_feature], opacity=0.7, nbinsx=50, name="Negative

    fig.add_trace(all_data_hist, row=1, col=1)
    fig.add_trace(positive_data_hist, row=2, col=1)
    fig.add_trace(negative_data_hist, row=3, col=1)

    # Update subplot layout
    fig.update_layout(
        showlegend=False,
        title_text=f"Histograms for {selected_feature}",
        height=500
    )

    return fig

# Run the app
if __name__ == '__main__':
    app.run_server(debug=True, port=8052)
```

Featured Histograms for Breast Cancer Dataset

mean radius



1. **Mean Radius**: Mean distance from the center to points on the tumor's perimeter. Shows a bimodal distribution, with larger radii associated with malignant tumors, indicating potential for classification.
2. **Mean Texture**: Mean grayscale values of pixels in an image. Overlapping values make it less effective for classification.
3. **Mean Perimeter**: Mean size of the core tumor area. Well-separated distribution, especially for larger perimeters, suggests potential for classification.
4. **Mean Area**: Mean area of the tumor. Positively skewed, indicating benign tumors tend to have smaller areas, while malignant tumors exhibit greater variability.
5. **Mean Smoothness**: Mean local variation in radius lengths. Complete overlap, offering limited discrimination between classes.
6. **Mean Compactness**: Mean of (perimeter^2 / area - 1.0). Slightly skewed and overlapping, providing partial separation.
7. **Mean Concavity**: Mean severity of concave portions of the contour. Positively skewed, slight overlap, and partial separation, especially for higher values.
8. **Mean Concave Points**: Mean number of concave portions of the contour. Positive skewness and overlap, indicating limited discriminatory power.
9. **Mean Symmetry**: Mean symmetry of the tumor. Overlapping, but positive instances have a higher peak, suggesting some value for classification.
10. **Mean Fractal Dimension**: Mean "coastline approximation" - 1. Overlapping entirely, offering limited classification information.
11. **Radius Error**: Standard error of the mean distances from the center to points on the perimeter. Positively skewed, slight overlap, and potential for separation.
12. **Texture Error**: Standard error of grayscale values. Overlapping values provide limited classification information.
13. **Perimeter Error**: Standard error of the tumor perimeter. Positively skewed, slight overlap, and potential for separation, with a bimodal distribution.
14. **Area Error**: Standard error of the tumor area. Positively skewed, slight overlap, and potential for separation.
15. **Smoothness Error**: Standard error of local variation in radius lengths. Overlapping values offer limited discrimination.
16. **Compactness Error**: Standard error of (perimeter^2 / area - 1.0). Some overlap, particularly above 0.1, suggests potential classification value.
17. **Concavity Error**: Standard error of the severity of concave portions of the contour. Positively skewed, slight overlap, and potential for separation.
18. **Concave Points Error**: Standard error of the number of concave portions of the contour. Slight overlap and potential for separation.
19. **Symmetry Error**: Standard error of tumor symmetry. Overlapping values with limited classification information.
20. **Fractal Dimension Error**: Standard error of the "coastline approximation" - 1. Positively skewed, slight overlap, and potential for separation.
21. **Worst Radius**: Worst (largest) mean distances from the center to points on the perimeter. Well-separated values, particularly for larger radii.
22. **Worst Texture**: Worst (largest) grayscale values. Overlapping values in the center of the curve with potential for separation at extremes.
23. **Worst Perimeter**: Worst (largest) tumor perimeter. Well-separated values, suggesting strong potential for classification.
24. **Worst Area**: Worst (largest) tumor area. Slightly overlapping values with a spread to the right, indicating potential for separation.
25. **Worst Smoothness**: Worst (largest) local variation in radius lengths. Bell curve distribution with some overlap among values.
26. **Worst Compactness**: Worst (largest) of (perimeter^2 / area - 1.0). Slight overlap with the negative instance spread to the right, suggesting potential for classification.
27. **Worst Concavity**: Worst (largest) severity of concave portions of the contour. Slightly overlapping values with the negative instance spread to the right.
28. **Worst Concave Points**: Worst (largest) number of concave portions of the contour. Slightly overlapping values with potential for separation.
29. **Worst Symmetry**: Worst (largest) tumor symmetry. Highly overlapping values with slight separation.
30. **Worst Fractal Dimension**: Worst (largest) "coastline approximation" - 1. Highly overlapping values with limited discriminatory power.

In summary, several features show potential for distinguishing between benign and malignant tumors, while others may require further analysis or feature engineering to improve their discriminative power in the classification task.

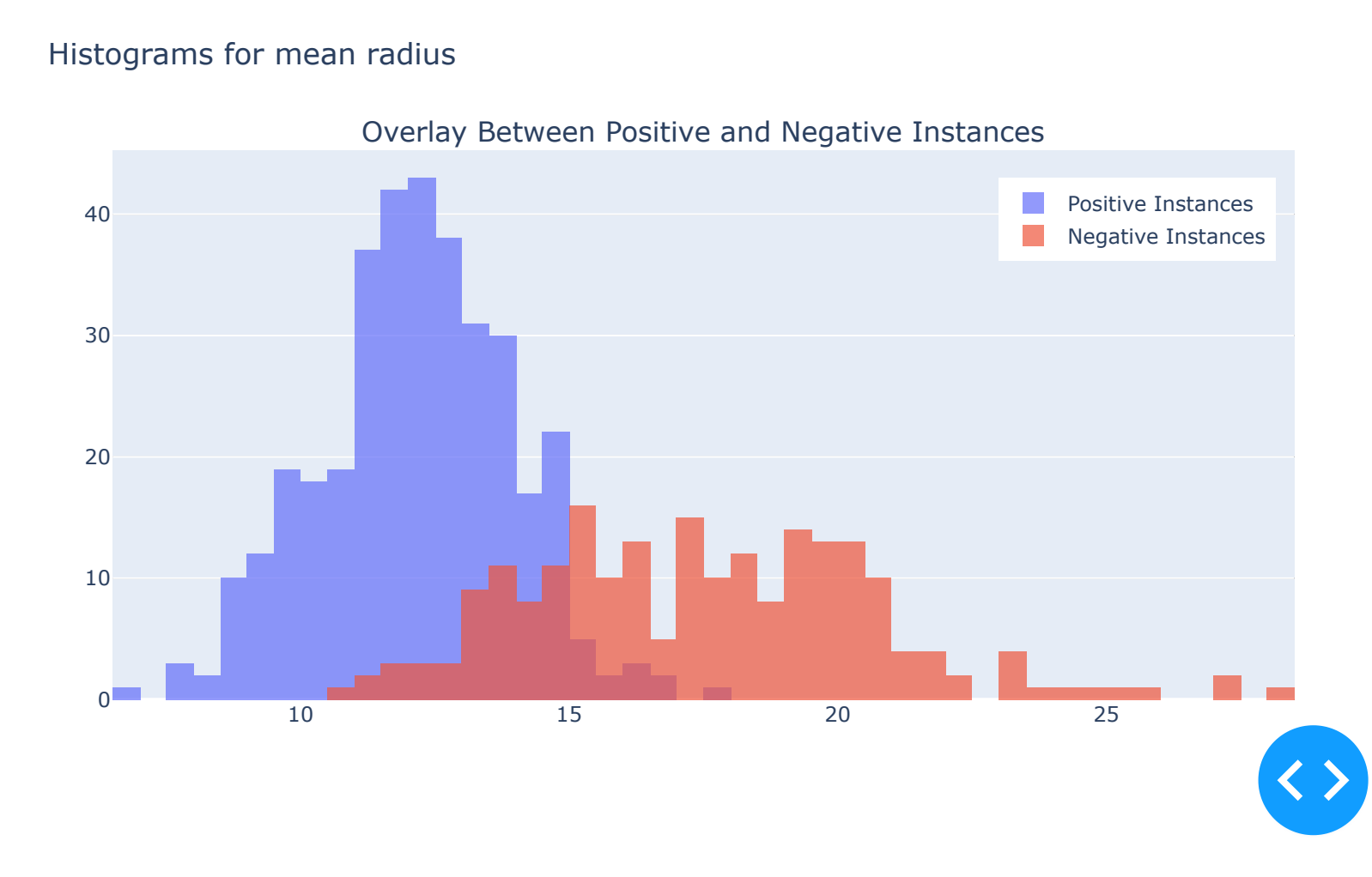
3. In your tool from #2, can you find a way to overlay the last two histograms of #2 on the same axes?

In [4]:

```
app_overlay = dash.Dash(__name__)

app_overlay.layout = html.Div([
    html.H1("Feature Histograms for Breast Cancer Dataset"),
    dcc.Dropdown(
        id='feature-selector-overlay',
        options=[{'label': col, 'value': col} for col in df.columns[:-1]],
        value=df.columns[0],
    ),
    dcc.Graph(id='histogram-subplot-overlay')
])

@app_overlay.callback(
    Output('histogram-subplot-overlay', 'figure'),
    Input('feature-selector-overlay', 'value')
)
def update_histogram_overlay(selected_feature):
    fig = make_subplots(rows=1, cols=1, subplot_titles=["Overlay Between Positive and Negative Instances"])

    fig.add_trace(go.Histogram(x=df[df['target'] == 1][selected_feature], opacity=0.7, nbinsx=50, name="Positive Instan
    fig.add_trace(go.Histogram(x=df[df['target'] == 0][selected_feature], opacity=0.7, nbinsx=50, name="Negative Instan

    # Update subplot layout
    fig.update_layout(
        showlegend=True,
        legend=dict(x=0.75, y=0.95),
        title_text=f"Histograms for {selected_feature}",
        height=500,
        barmode="overlay"
    )

    # Update bin settings to ensure proper overlay
    bin_settings = dict(
        start=min(df[selected_feature].min(), df[df['target'] == 1][selected_feature].min(), df[df['target'] == 0][sele
        end=max(df[selected_feature].max(), df[df['target'] == 1][selected_feature].max(), df[df['target'] == 0][select
        size=0.1  # Adjust bin size as needed
    )

    return fig

if __name__ == '__main__':
    app_overlay.run_server(debug=True, port=8053)
```

Feature Histograms for Breast Cancer Dataset

mean radius



In this code block, we define a callback function in a Dash app that updates a histogram subplot when a specific feature is selected. It overlays histograms for positive and negative instances of the selected feature, allowing users to visualize the distribution of data.

4. In your tool from #2, play with the interactive controls offered by Plotly and comment on their usefulness on inspecting histograms of the particular data set.

- **Zoom and Pan:** We can zoom in to get a closer look at a specific part of the plot or pan to navigate across the data. This is handy when dealing with large datasets.
- **Hover Details:** When we hover over a data point, we'll see a tooltip with information about that point, such as its value or position. This helps in getting precise information.
- **Legend Control:** Clicking on items in the legend can show or hide specific data series. This is useful for isolating or comparing different parts of the data.
- **Linked Subplots:** If we have multiple subplots that share the same x-axis or y-axis, zooming or panning in one subplot will affect the others. It's great for comparing patterns across subplots.
- **Download as Image:** You have the option to download the plot as a static image (e.g., PNG, JPEG, SVG, or PDF). This is useful for sharing or including the plot in documents.
- **Spike Lines:** Spike lines are lines that extend across the axis and intersect with the hovered point. They help us align the point with axis ticks.
- **Data Comparison on Hover:** When we have multiple data series, hovering over points at the same x-position allows us to compare their values.
- **Autoscale and Reset:** After zooming or panning, we can either autoscale the plot to fit the data or reset the axes to their original state.

5. Using Dash, create a second interactive tool that allows the user to select a pair of features and displays a scatter plot where the two classes are coloured differently. Select ten pairs based on the insights you obtained from the histograms, and describe your observations. Hint: Use **scatter_plot or **scatter_matrix** in Plotly Express. Use the colour parameter to colour according to the class.**

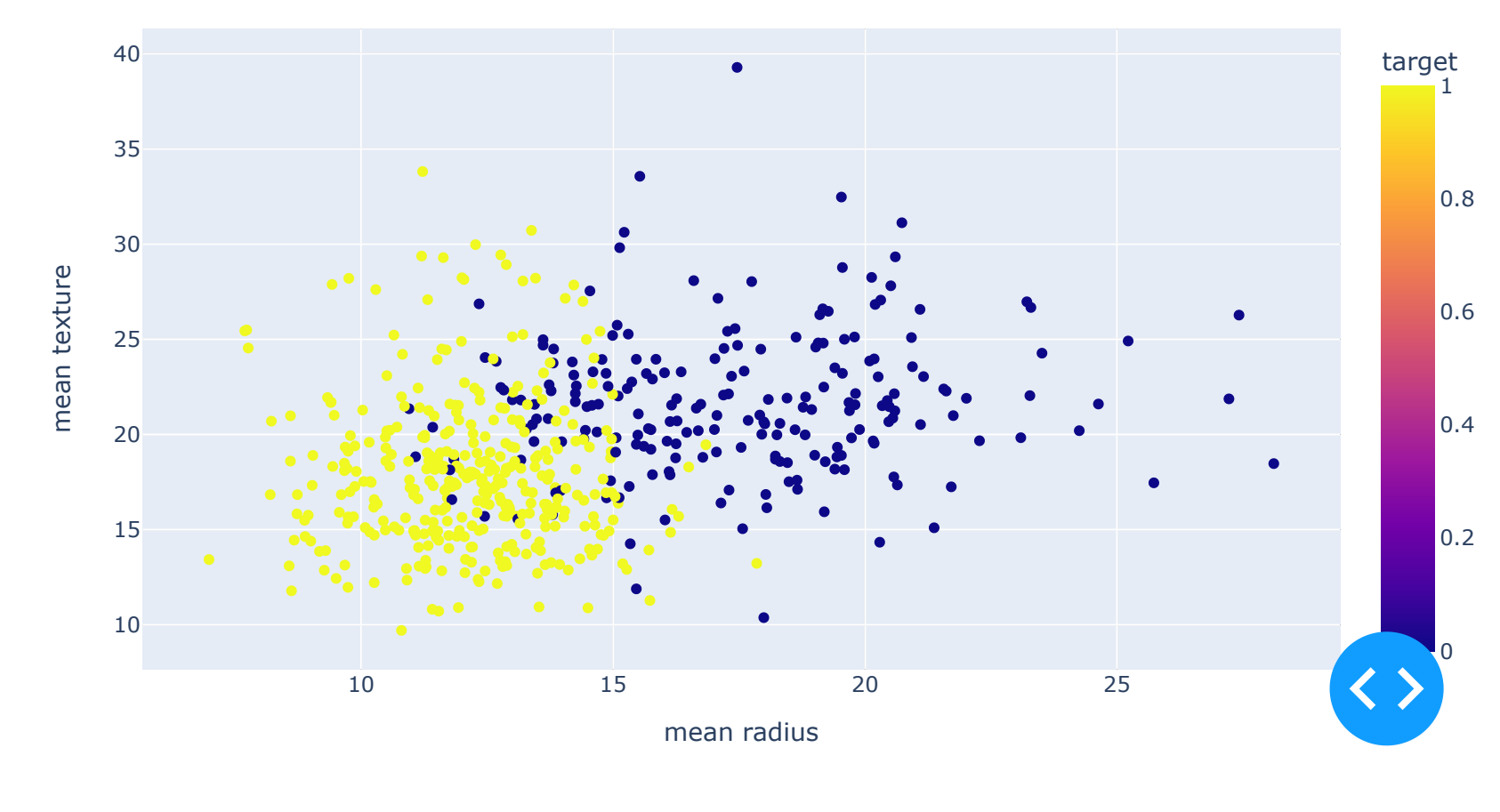
In [5]:

```
# Code here
scatter_plot_app = dash.Dash(__name__)

# Define the layout
scatter_plot_app.layout = html.Div([
    html.H1("Scatter Plot for Breast Cancer Dataset"),
    dcc.Dropdown(
        id='xaxis-feature',
        options=[{'label': col, 'value': col} for col in df.columns[:-1]],
        value=df.columns[0]
    ),
    dcc.Dropdown(
        id='yaxis-feature',
        options=[{'label': col, 'value': col} for col in df.columns[:-1]],
        value=df.columns[1]
    ),
    dcc.Graph(id='scatter-plot')
])

# Define the callback
@scatter_plot_app.callback(
    Output('scatter-plot', 'figure'),
    Input('xaxis-feature', 'value'),
    Input('yaxis-feature', 'value')
)
def update_scatter(xaxis_feature, yaxis_feature):
    fig = px.scatter(df, x=xaxis_feature, y=yaxis_feature, color='target')
    fig.update_layout(height=500)
    return fig

# Run the app2
if __name__ == '__main__':
    scatter_plot_app.run_server(debug=True, port = 8054)
```

Scatter Plot for Breast Cancer Dataset

mean radius

mean texture



10 pairs of features along with reasons for their potential utility and observations:

1. **Mean Radius and Mean Perimeter**: These features are likely correlated, as larger mean radii would result in larger mean perimeters. The combination of these features may help distinguish tumor size, which can be crucial for classification.
Observation: It shows some degree of separation between the classes, but it also has overlapping values, indicating that it may not be a perfect separator
2. **Mean Texture and Mean Smoothness**: Texture and smoothness might provide complementary information about the tumor's characteristics. Higher texture with lower smoothness could indicate malignancy.
Observation: It has mixed and scattered distribution of values with some overlap, making it a challenging pair for classification.
3. **Mean Perimeter and Mean Area**: A combination of perimeter and area could capture the tumor's overall shape. Malignant tumors might have larger perimeters for their respective areas, leading to a potential classification criterion.
Observation: Displays separation between the classes, but it also has some overlapping values and a few outliers.
4. **Mean Compactness and Mean Concavity**: Compactness and concavity could jointly represent how irregular and compact the tumor is. High compactness and high concavity could indicate malignancy.
Observation: It shows clear separation between the classes with some overlapping values, but the values are spread out.
5. **Mean Concavity and Mean Concave Points**: Combining these features can provide information on the severity and number of concave portions. Elevated values in both may signify malignant tumors.
Observation: It shows clear separation and some overlapping values, though the values are widely dispersed, which may affect classification.
6. **Mean Symmetry and Mean Smoothness**: The combination of symmetry and smoothness may capture the overall regularity of the tumor's surface. Lower symmetry and smoothness could suggest malignancy.
Observation: It has mixed and scattered distribution of values with some overlap, making it a challenging pair for classification.
7. **Worst Radius and Worst Perimeter**: These features represent the largest values for radius and perimeter, respectively. Larger values for both could indicate malignant tumors.
Observation: There is separation between classes, but some overlapping values and outliers are present, particularly among higher values.
8. **Worst Texture and Worst Smoothness**: Combining the worst texture and worst smoothness may capture the extreme values in texture and smoothness. Higher worst texture and lower worst smoothness could be indicative of malignancy.
Observation: It has mixed and scattered distribution of values with some overlap, making it a challenging pair for classification.
9. **Worst Perimeter and Worst Area**: The largest perimeter combined with the largest area may provide insight into the overall size and shape of the tumor. High values for both could signal malignancy.
Observation: It has clear separation between classes and might be highly useful for classification.
10. **Worst Compactness and Worst Concavity**: Combining worst compactness and worst concavity may offer information on the irregularity and compactness of the tumor's concave portions. Higher values for both features might be associated with malignancy.
Observation: It shows clear separation between the classes, but it also has some overlapping values and a few outliers.

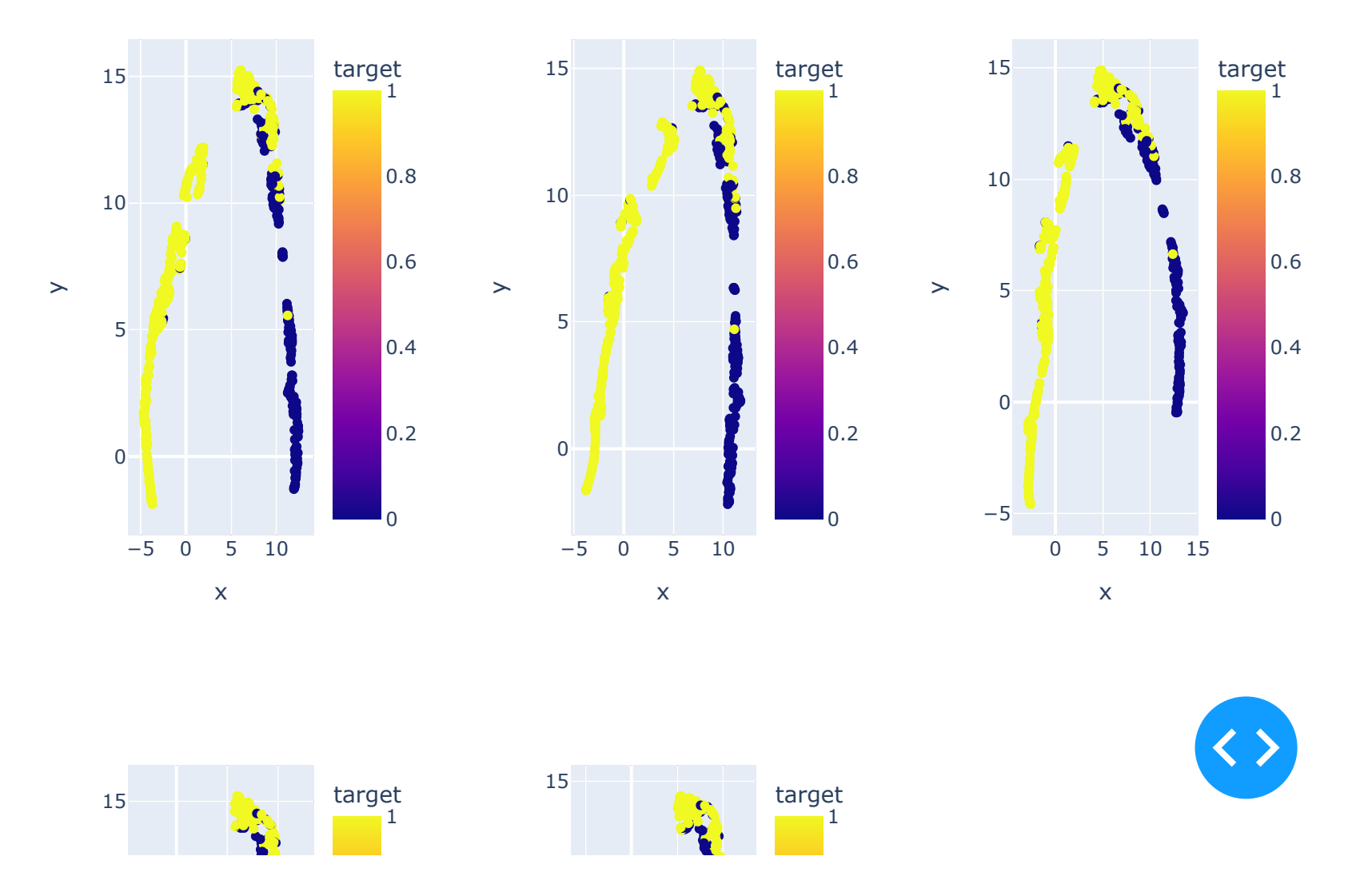
These feature pairs are selected based on the assumption that they might collectively capture different aspects of tumor characteristics, such as size, shape, regularity, and irregularity, which are crucial for distinguishing between benign and malignant tumors. However, thorough feature selection and model training are necessary to confirm their actual significance in a classification task.

6. Familiarize yourself with UMAP, and project the data set into two dimensions. Using Dash, display the projected data as a scatterplot, where the two classes are coloured differently. Comment on how distinct the two classes are in the scatterplot. Repeat for five different UMAP projections. Comment on the visual variability of the results. Discuss similarities and differences with t-SNE from assignment 1.

In [6]:

```
# Code here
# Prepare the data
reducer = umap.UMAP()
embeddings = []
for i in range(5):
    embedding = reducer.fit_transform(df[df.columns[:-1]])
    # Create a new DataFrame for the projected data
    projected_df = pd.DataFrame(embedding, columns=['x', 'y'])
    projected_df['target'] = df['target']
    embeddings.append(projected_df)

app3 = dash.Dash(__name__)

# The 'width' is set to 33% for each graph to evenly distribute them side by side,
# optimizing the layout for clear visualization.

# We used width as 33% because while converting to PDF,
# it was not showing all the graphs that were hidden because of scroll

# Please note that this specific styling will be removed when submitting the code
# in an ipynb format, ensuring a clean and unaltered representation of the graphs
app3.layout = html.Div([
    html.H1("UMAP Projection of Breast Cancer Dataset"),
    dcc.Graph(id='scatter-plot-0', style={'width': '33%', 'display': 'inline-block'}),
    dcc.Graph(id='scatter-plot-1', style={'width': '33%', 'display': 'inline-block'}),
    dcc.Graph(id='scatter-plot-2', style={'width': '33%', 'display': 'inline-block'}),
    dcc.Graph(id='scatter-plot-3', style={'width': '33%', 'display': 'inline-block'}),
    dcc.Graph(id='scatter-plot-4', style={'width': '33%', 'display': 'inline-block'})
])

def update_scatter(id, projection_index):
    projected_df = embeddings[projection_index]
    fig = px.scatter(projected_df, x='x', y='y', color='target')
    return fig

@app3.callback(
    Output('scatter-plot-0', 'figure'),
    Input('scatter-plot-0', 'id')
)
def update_scatter_0(id):
    return update_scatter(id, 0)

@app3.callback(
    Output('scatter-plot-1', 'figure'),
    Input('scatter-plot-1', 'id')
)
def update_scatter_1(id):
    return update_scatter(id, 1)

@app3.callback(
    Output('scatter-plot-2', 'figure'),
    Input('scatter-plot-2', 'id')
)
def update_scatter_2(id):
    return update_scatter(id, 2)

@app3.callback(
    Output('scatter-plot-3', 'figure'),
    Input('scatter-plot-3', 'id')
)
def update_scatter_3(id):
    return update_scatter(id, 3)

@app3.callback(
    Output('scatter-plot-4', 'figure'),
    Input('scatter-plot-4', 'id')
)
def update_scatter_4(id):
    return update_scatter(id, 4)

# Run the app3
if __name__ == '__main__':
    app3.run_server(debug=True, port = 8055)
```

UMAP Projection of Breast Cancer Dataset



Note : The 'width' is set to 33% for each graph to evenly distribute them side by side, optimizing the layout for clear visualization.

We used width as 33% because while converting to PDF, it was not showing all the graphs that were hidden because of scroll

Please note that this specific styling will be removed when submitting the code in an ipynb format, ensuring a clean and unaltered representation of the graphs

Comment on how distinct the clusters are:

1. It appears that the two classes are quite distinct. The points from each class form separate clusters, with minimal overlap. This suggests that the UMAP projection has successfully captured the structure of the data in a way that separates the classes.
2. We could conclude that, as value increases on the `X` axis the probability of a cell being malignant (cancerous) decreased. The lower the value was the higher the chances were of a cell being cancerous.

Comment on the visual variability of the results:

1. Although the shape of clusters in each iteration changes slightly, its important that the structure and relationship between clusters and points are preserved properly, which is clearly evident from looking at all the 5 graphs.
2. When it comes to the differences between the overall shape of the clusters, it could be due to the fact that UMAP is a stochastic algorithm which incorporates random elements such as stochastic gradient descent which is used in optimizing the low dimensional graph, etc.

Differences between UMAP and T-SNE:

1. The first difference is how the low dimensional graph is formulated. In T-SNE the points (scatter points) are randomly arranged on a straight line, whereas in UMAP the initialization of low dimensional graph is based on spectral embedding of the scatter points. It means to say that if we were to perform T-SNE on a dataset multiple times, we would end up with different low-dimensional graphs each time, but with UMAP the low dimensional graph would be the same each time.
2. In T-SNE's iteration to order the low dimensional graph, all the points need to move at once, whereas in UMAP can either move 1 point, or a subset of n number of points. This makes UMAP helpful for large datasets. T-SNE is computationally expensive and may not scale well for large datasets.
3. The way similarity scores are calculated in both T-SNE and UMAP are different. In UMAP, the similarity scores are calculated by considering n-nearest neighbors, wheras in T-SNE it is calculated using the distance between clusters.
4. In terms of visualization, T-SNE cannot accurately represent the distance between clusters, i.e., it places the clusters on the low dimensional graph far away from where it was supposed to be in the higher dimensional graph. In UMAP, we can accurately visualize the distance (boundaries) between clusters in the low dimensional graphs.
5. In comparison to the T-SNE graph from assignment-1, it was clear that UMAP was able to scale the distance (boundary) between the two clusters better than T-SNE.
6. The data points in the T-SNE image are scattered around the center of the plot, with one cluster being more spread out than the other. In contrast, the data points in the UMAP image form two distinct lines with positive and negative slopes.

REFERENCES:

[1] Join conversation. (n.d.). Microsoft Teams. Retrieved October 7, 2023, from https://teams.microsoft.com/l/message/19:5adb377c-c9b3-4e51-957c-deb93c053354_7f2568ef-46e4-430d-9cc7-daf96d1ef6da@unq.gbl.spaces/1696647152861?context=%7B%22contextType%22%3A%22chat%22%7D
[2] Starmer, S. W. J. [@statquest]. (2022, March 7). UMAP Dimension Reduction, Main Ideas!!! Youtube. https://www.youtube.com/watch?v=eN0wFzBA4Sc
[3] UMAP API Guide — umap 0.5 documentation. (n.d.). Readthedocs.Io. Retrieved October 7, 2023, from https://umap-learn.readthedocs.io/en/latest/api.html

In [ ]:

```
```