TASK #1: PLOT INTERACTIVE BOX PLOT USING PLOTLY EXPRESS

In [2]:

```
# The plotly Python package empowers anyone to create, manipulate and render graphical figures.
# The figures are represented by data structures referred to as figures.
# The rendering process uses the Plotly.js JavaScript library under the hood but you never need to
use Java directly.
# Figures can be represented in Python either as dictionaries or as instances of the plotly.graph_
objects

# Note:
# Plotly Express is the recommended entry-point into the plotly package
# Plotly Express is the high-level plotly.express module that consists of Python functions which r
eturn fully-populated plotly.graph_objects.Figure objects.
# plotly.express module contains functions that can create interactive figures using a very few li
nes of code
# Plotly Express is refered to as px.
# Plotly Express is a built-in part of the plotly library
# Plotly Express function uses graph objects internally and returns a plotly.graph_objects.Figure
instance.
# check out the documentation here: https://plotly.com/python/plotly-express/



# A box plot is a statistical representation of numerical data through their quartiles.
# The ends of the box represent the lower and upper quartiles, while the median (second quartile)
is marked by a line inside the box.

import plotly.express as px
import pandas as pd
import numpy as np
```

In [3]:

```
# Import Cancer data drom the Sklearn library
cancer_df = pd.read_csv("cancer.csv")
```

In [4]:

```
# Check out the head of the dataframe
cancer_df.head(5)
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	w perim
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	17.33	184.60
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	23.41	158.80
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	25.53	152.50
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	26.50	98.87
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	16.67	152.20

5 rows × 31 columns

In [5]:

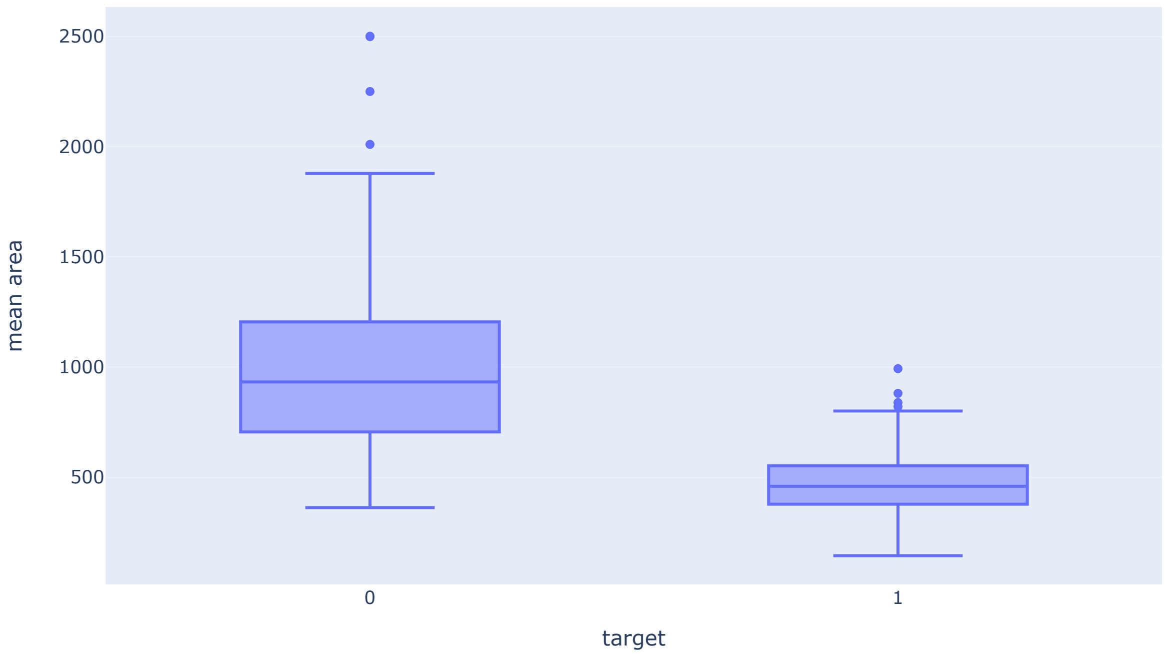
```
# Check out the tail of the dataframe
cancer_df.tail()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	per
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623	...	26.40	166.
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533	...	38.25	155.
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648	...	34.12	126.
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016	...	39.42	184.
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884	...	30.37	59.1

5 rows × 31 columns

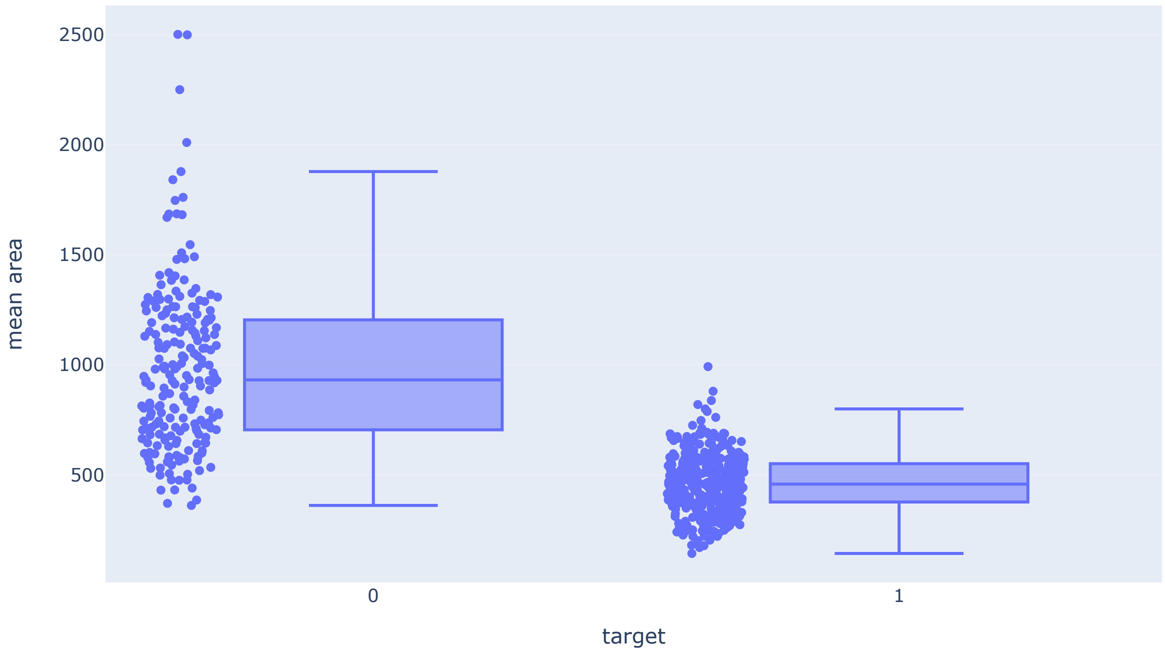
In [6]:

```
fig = px.box(cancer_df, x = 'target', y = 'mean area')
fig.show()
```



In [16]:

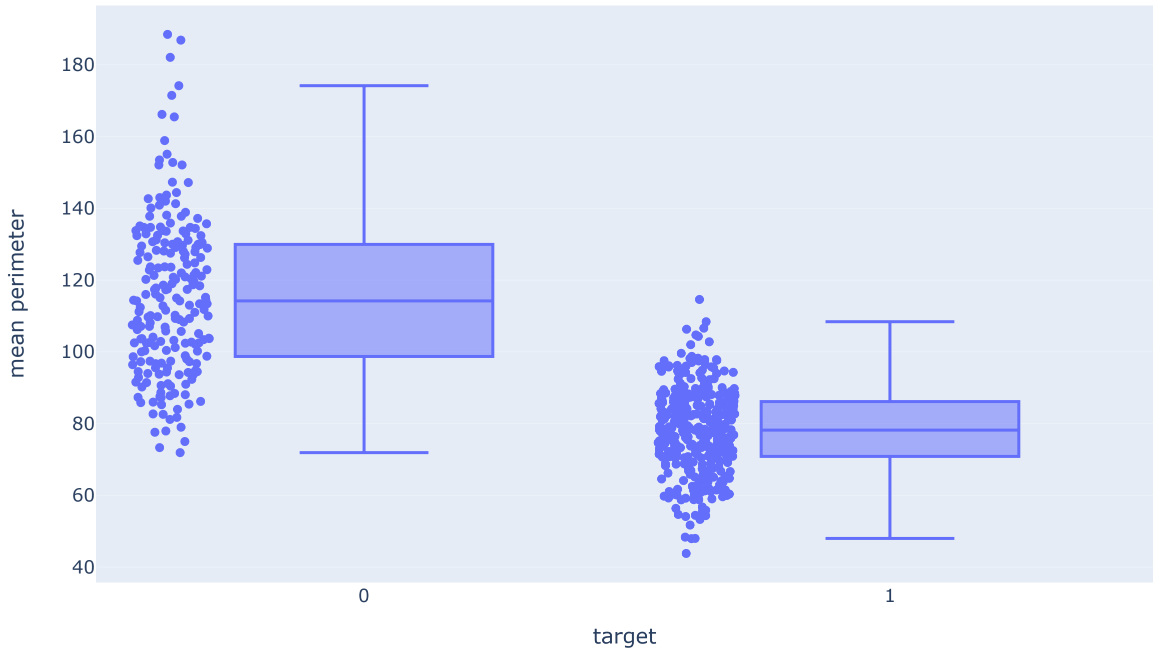
```
fig = px.box(cancer_df, x = 'target', y = 'mean area', points = 'all')
fig.show()
```



MINI CHALLENGE #1:

- **Plot the boxplot for Mean Perimeter, use points = "all"**

In [9]:
```
fig = px.box(cancer_df, x = "target", y = "mean perimeter", points = "all")
fig.show()
```



TASK #2: PLOT INTERACTIVE HISTOGRAMS

In [8]:
```
cancer_df
```

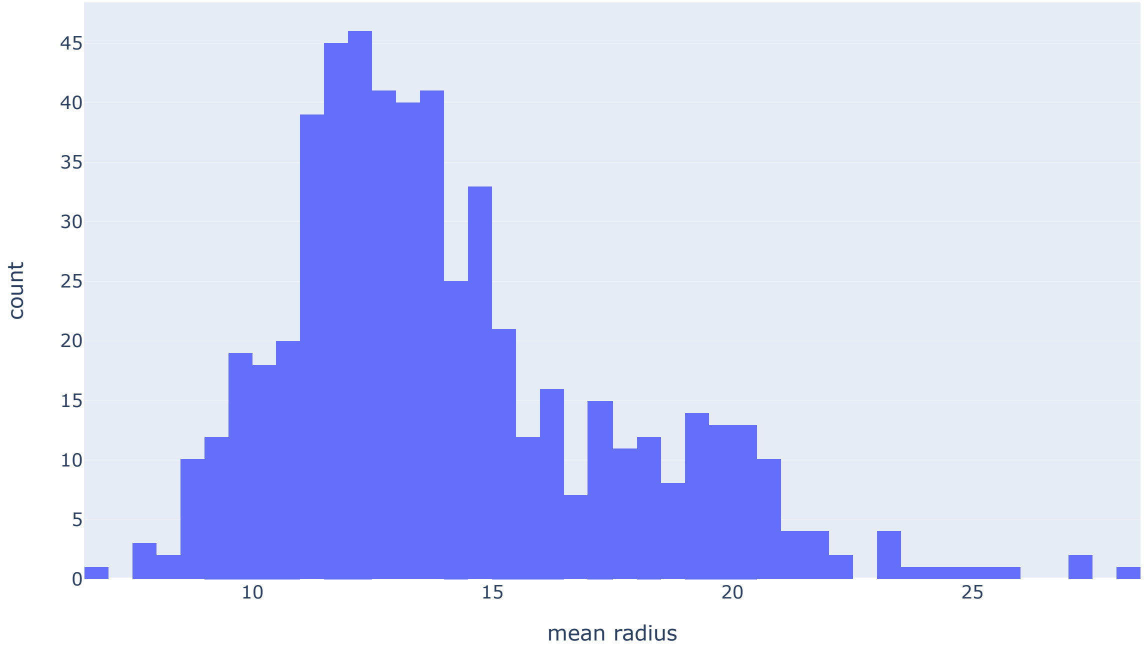
	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	per
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	0.07871	...	17.33	184.
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.05667	...	23.41	158.
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.05999	...	25.53	152.
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.09744	...	26.50	98.8
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.05883	...	16.67	152.
...	...	...	...	...	...	...	...	...	...	...	...	...	...
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623	...	26.40	166.
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533	...	38.25	155.
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648	...	34.12	126.
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016	...	39.42	184.
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884	...	30.37	59.1

```
569 rows × 31 columns
```

In [9]:

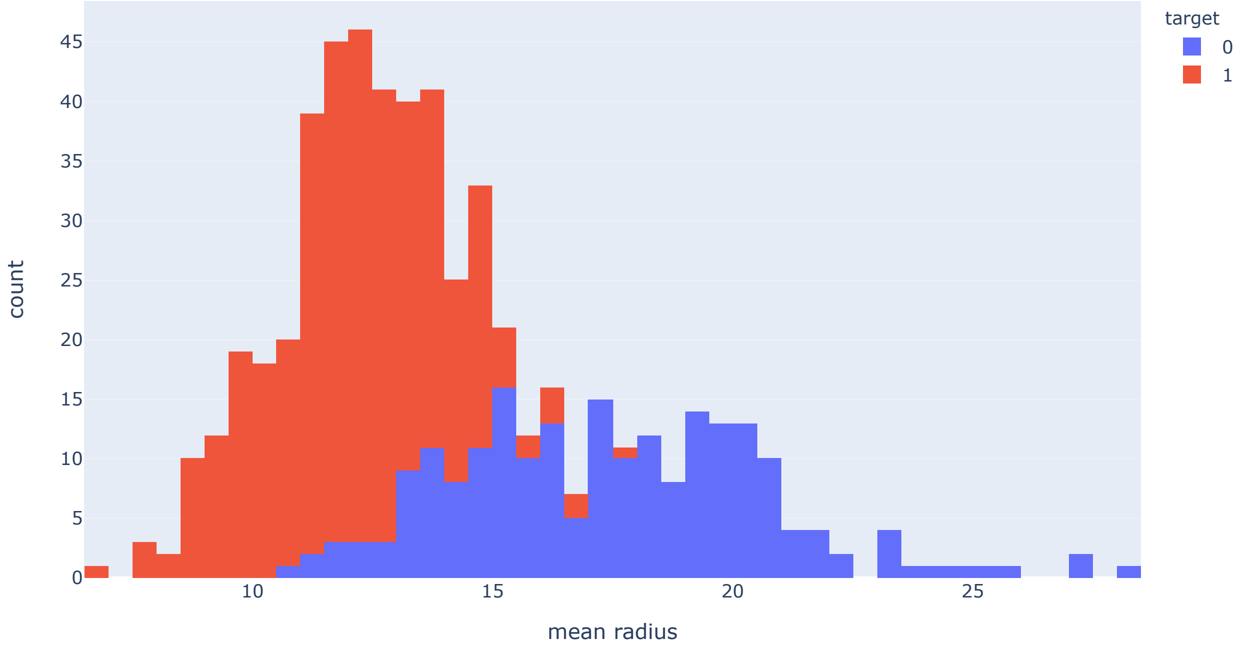
```
# A histogram is representation of the distribution of numerical data, where the data are binned and the count for each bin is represented.

fig = px.histogram(cancer_df, x = 'mean radius', nbins = 60)
fig.show()
```



In [10]:

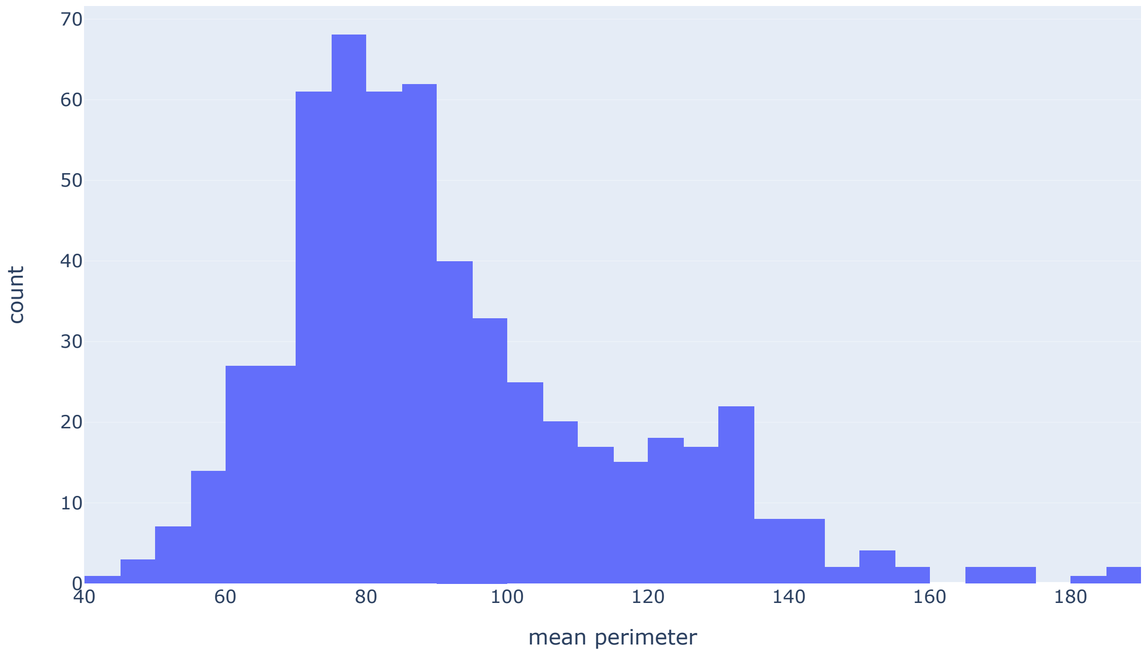
```
fig = px.histogram(cancer_df, x = 'mean radius', color = 'target', nbins = 60)
fig.show()
```



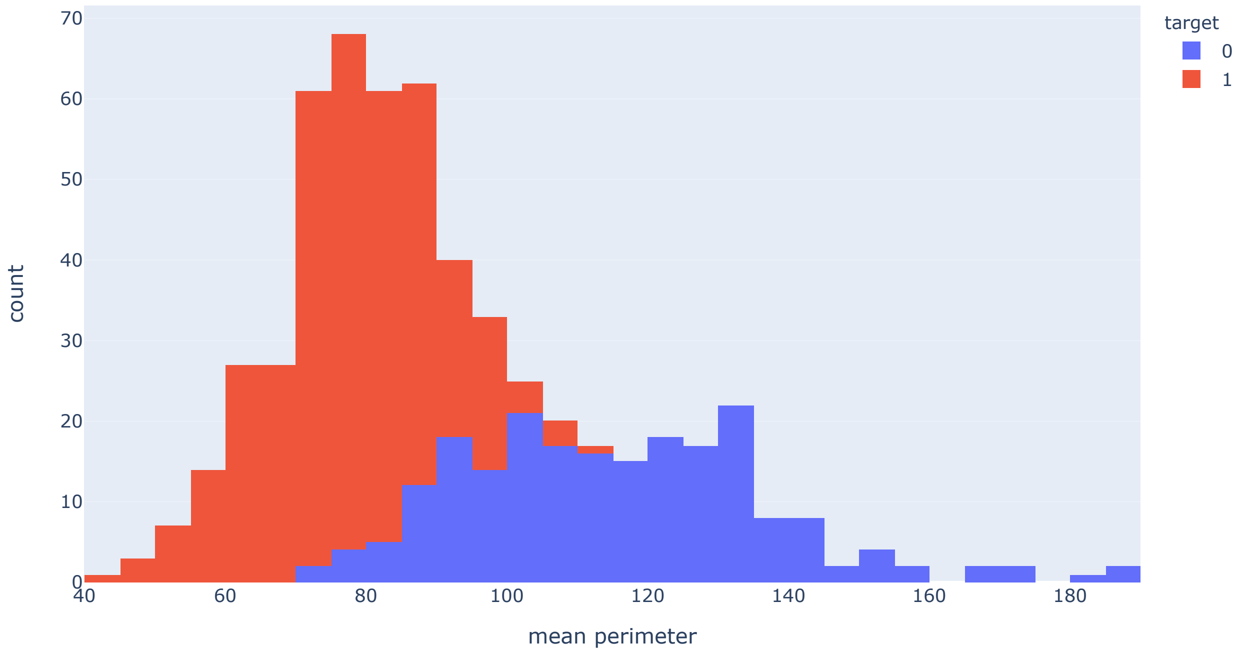
MINI CHALLENGE #2:

- **Plot the histogram for the mean permieter for the entire dataset**
- **Plot the histogram for the mean perimeter for each of the class independantly**

In [12]:
```
fig = px.histogram(cancer_df, x = 'mean perimeter', nbins = 60)
fig.show()
```



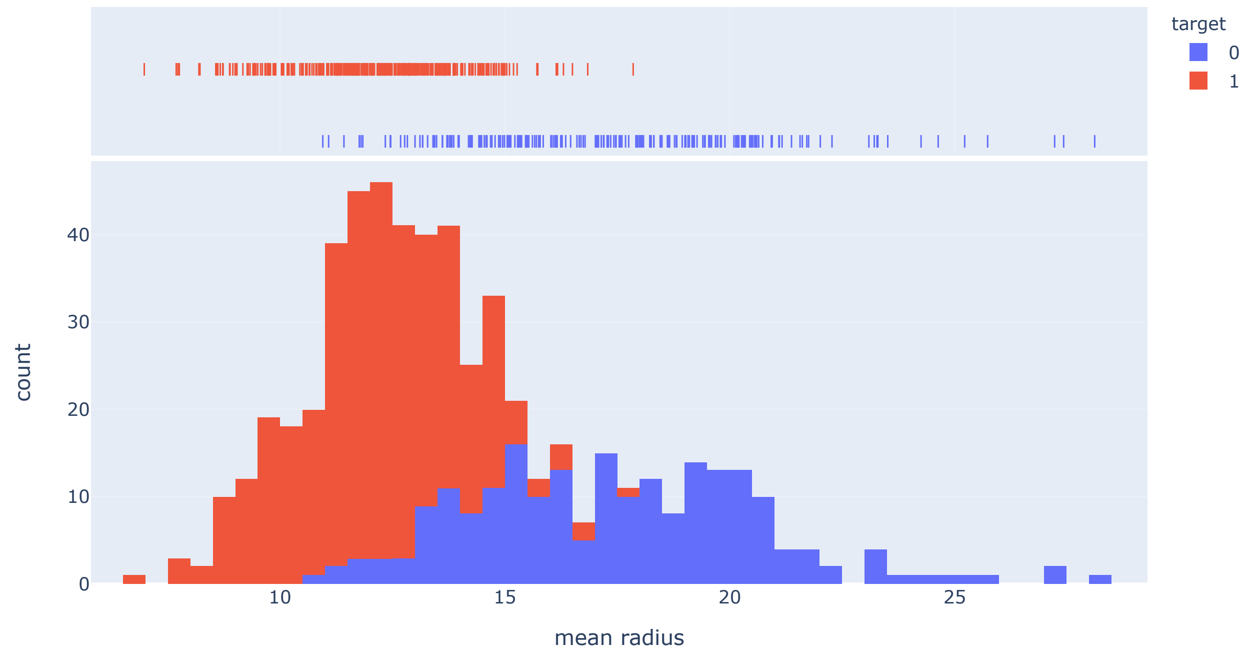
In [14]:
```
fig = px.histogram(cancer_df, x= 'mean perimeter', color = 'target', nbins = 60)
fig.show()
```



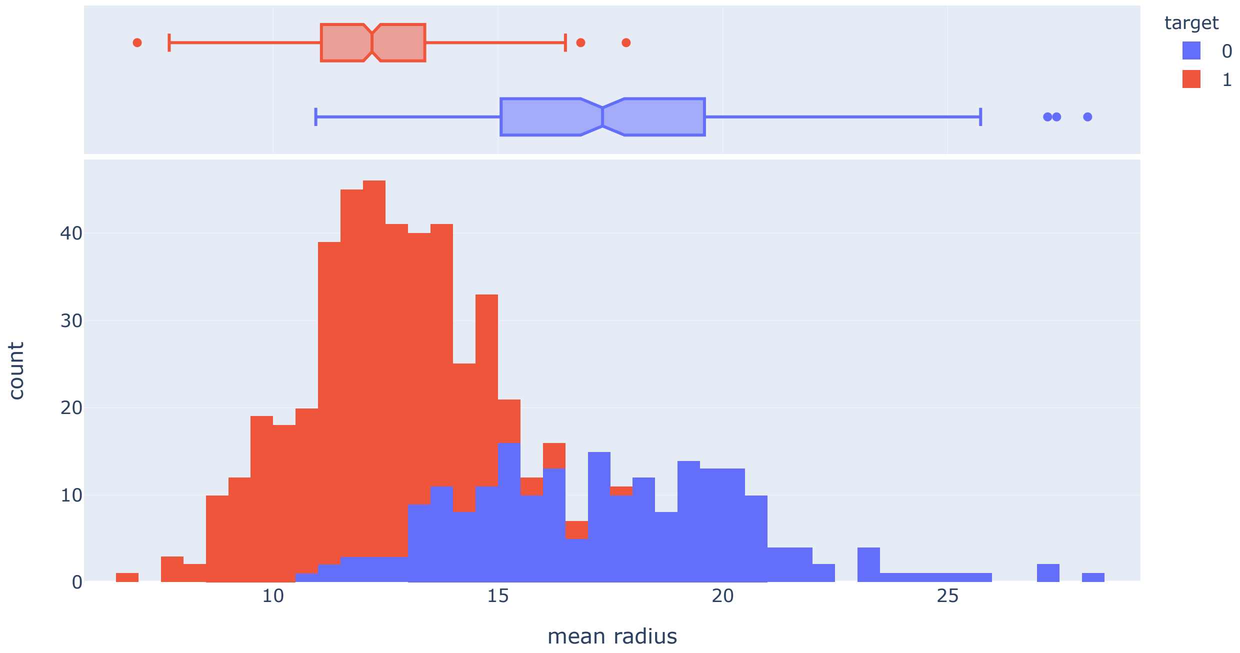
TASK #3: PLOT INTERACTIVE HISTOGRAMS WITH MARGINAL PLOTS

In [15]:

```
# With the marginal keyword, a subplot is drawn alongside the histogram, visualizing the distribution.
# For example, the plotly.express function px.histogram can add a subplot with a different statistical representation than the histogram, given by the parameter marginal. Plotly Express is the easy-to-use, high-level interface to Plotly,
fig = px.histogram(cancer_df, x = 'mean radius', color = 'target', marginal = 'rug', nbins = 60)
fig.show()
```



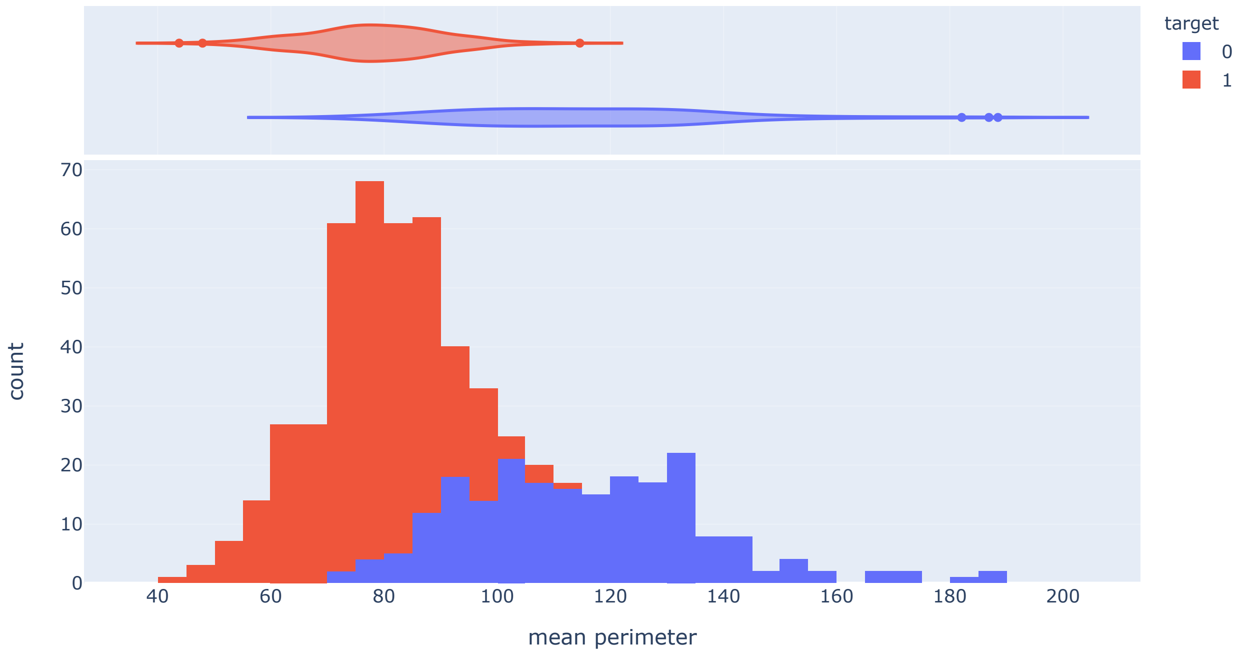
```
In [17]: fig = px.histogram(cancer_df, x = 'mean radius', color = 'target', marginal = 'box', nbins = 60)
         fig.show()
```



MINI CHALLENGE #3:

- **Plot the histogram for the mean perimeter using 40 bins and explore a new marginal plot**

In [18]:
```
fig = px.histogram(cancer_df, x = 'mean perimeter', color = 'target', marginal = 'violin', nbins = 
40)
fig.show()
```

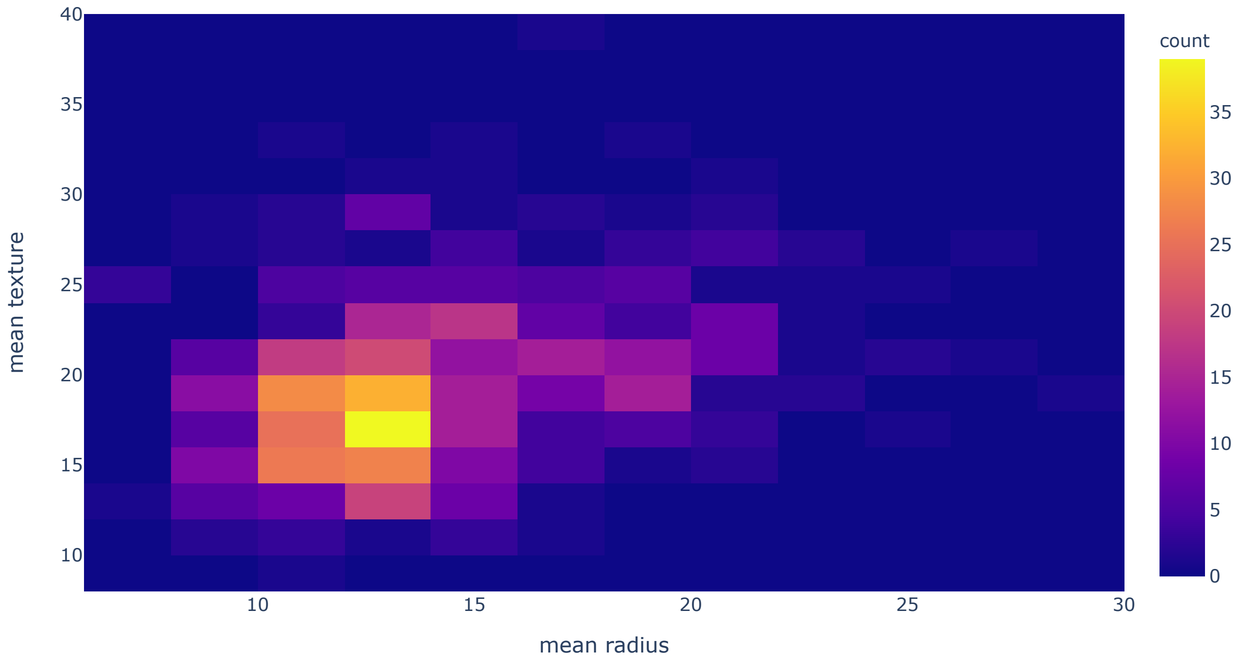


TASK #4: PLOT INTERACTIVE DENSITY MAP

In [19]:

```python
# A 2D histogram, also known as a density heatmap, is the 2-dimensional generalization of a histogram which resembles a heatmap but is computed by grouping a set of points specified by their x and y coordinates into bins,
# and applying an aggregation function such as count or sum (if z is provided) to compute the color of the tile representing the bin.
# This kind of visualization (and the related 2D histogram contour, or density contour) is often used to manage over-plotting, or situations where showing large data sets as scatter plots would result in points overlapping each other and hiding patterns.

fig = px.density_heatmap(cancer_df, x = 'mean radius', y = 'mean texture')
fig.show()
```



In [20]:

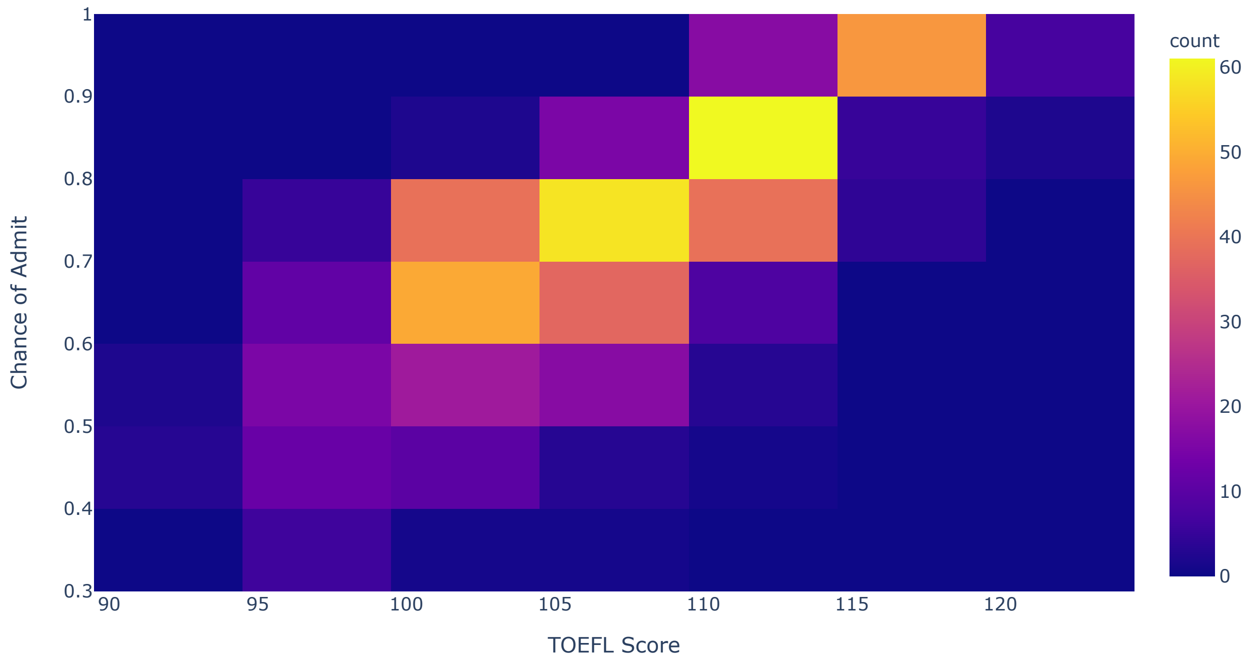
```python
# read univeristy_admission.csv dataset
university_df = pd.read_csv('university_admission.csv')
university_df
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65
...	...	...	...	...	...	...	...	...	...
495	496	332	108	5	4.5	4.0	9.02	1	0.87
496	497	337	117	5	5.0	5.0	9.87	1	0.96
497	498	330	120	5	4.5	5.0	9.56	1	0.93
498	499	312	103	4	4.0	5.0	8.43	0	0.73
499	500	327	113	4	4.5	4.5	9.04	0	0.84

500 rows × 9 columns

In [21]:

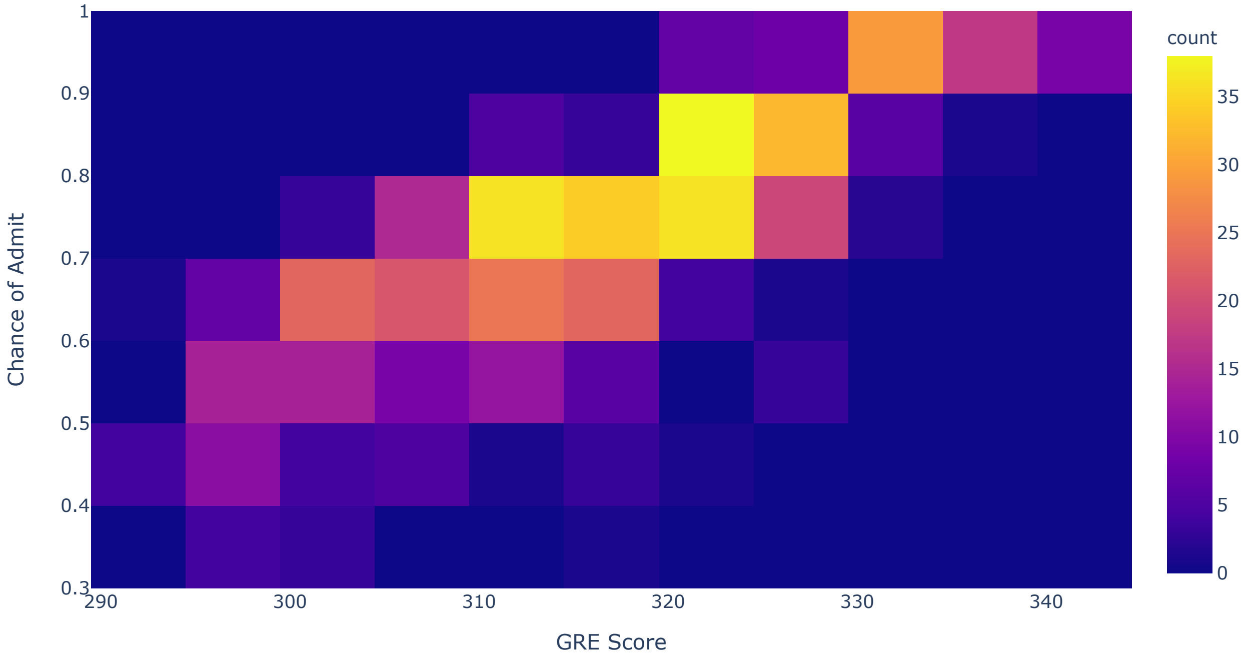
```
# plot density_heatmap
fig = px.density_heatmap(university_df, x = 'TOEFL Score', y = 'Chance of Admit')
fig.show()
```



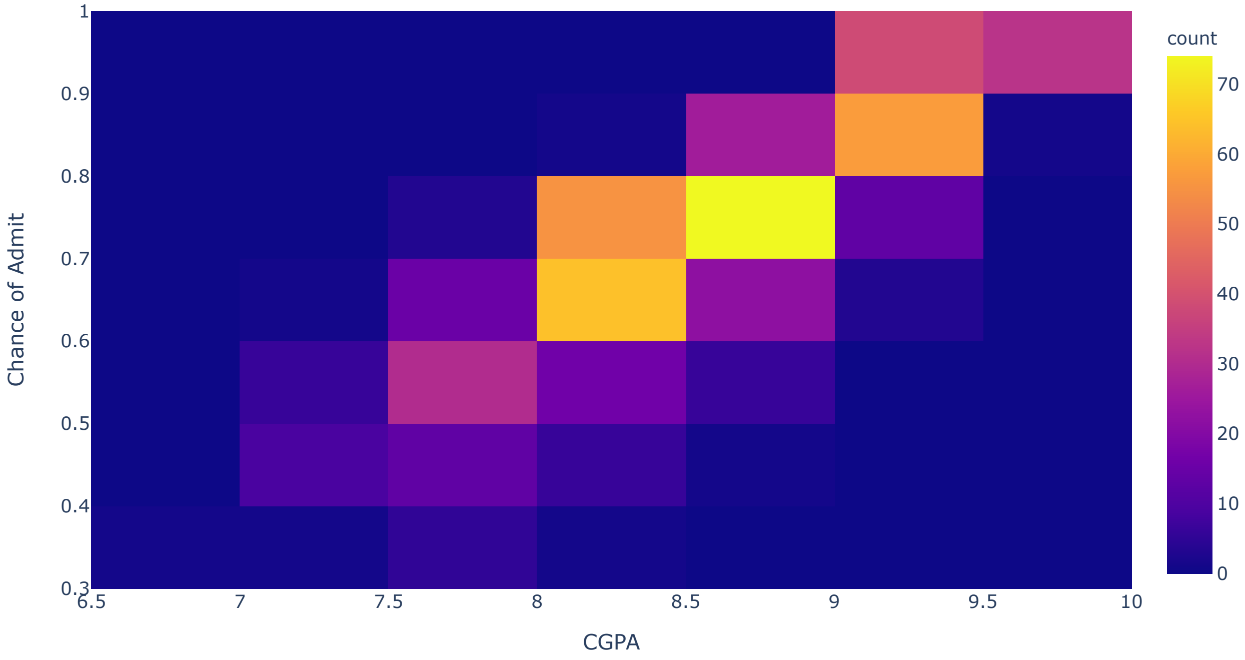
MINI CHALLENGE #4:

- **Plot density map between GRE Score vs. Chance of admission**
- **Plot density map between GPA vs. Chance of admission**

```
In [22]: fig = px.density_heatmap(university_df, x = 'GRE Score', y = 'Chance of Admit')
         fig.show()
```



```
In [24]: fig = px.density_heatmap(university_df, x = 'CGPA', y = 'Chance of Admit')
         fig.show()
```



```
In [ ]:
```

TASK #5: PLOT INTERACTIVE SCATTER MATRIX

In [25]:

```
university_df
```

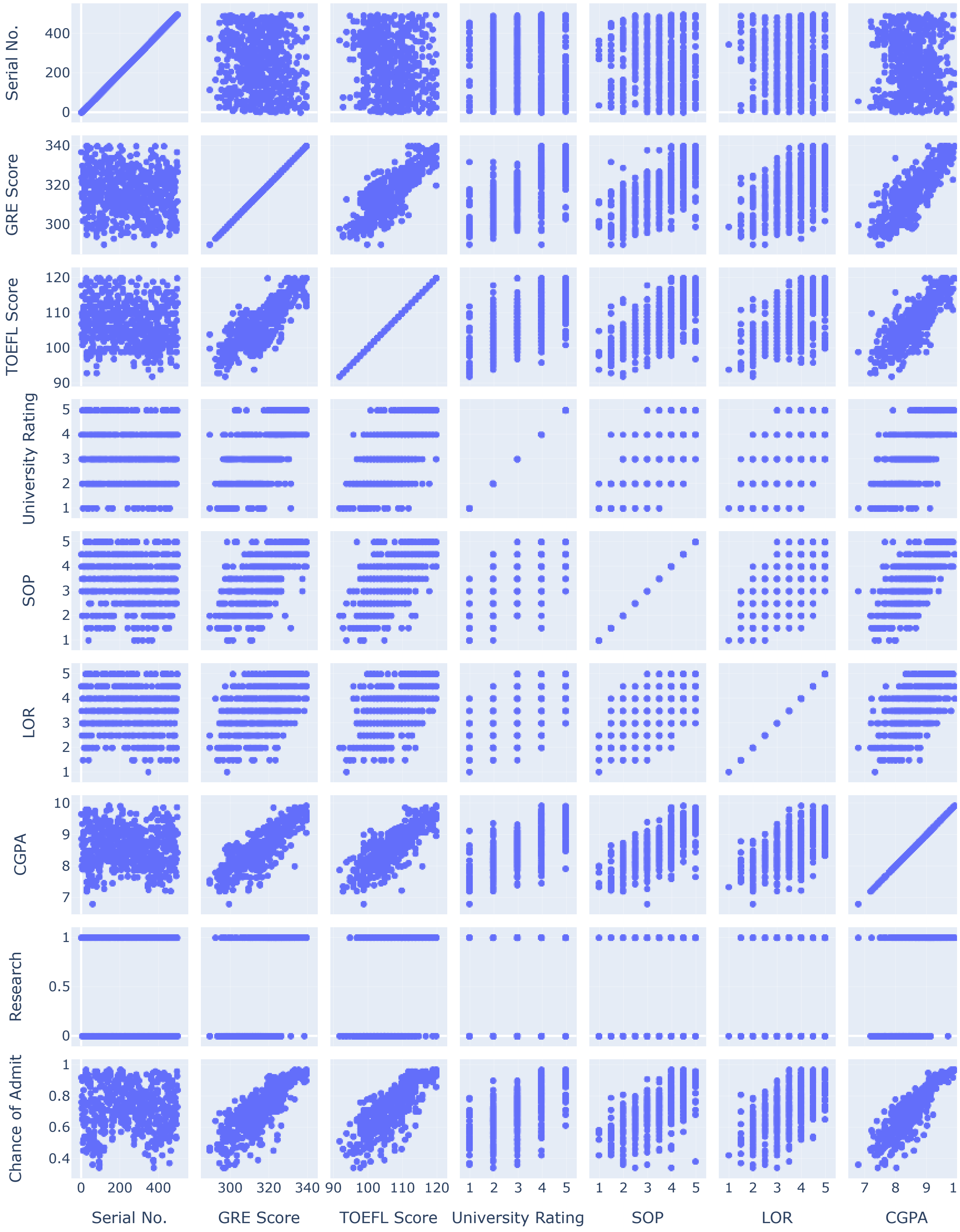
	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65
...	...	...	...	...	...	...	...	...	...
495	496	332	108	5	4.5	4.0	9.02	1	0.87
496	497	337	117	5	5.0	5.0	9.87	1	0.96
497	498	330	120	5	4.5	5.0	9.56	1	0.93
498	499	312	103	4	4.0	5.0	8.43	0	0.73
499	500	327	113	4	4.5	4.5	9.04	0	0.84

```
500 rows × 9 columns
```

In [26]:

```
# A scatterplot matrix is a matrix associated to n numerical arrays (data variables), $X_1,X_2,…,X_n$ , of the same length.
# The cell (i,j) of such a matrix displays the scatter plot of the variable Xi versus Xj.
# Here we show the Plotly Express function px.scatter_matrix to plot the scatter matrix for the columns of the dataframe. By default, all columns are considered.

fig = px.scatter_matrix(university_df, width = 1200, height = 1200)
fig.show()
```


Serial No.
400
200
0
GRE Score
340
320
300
TOEFL Score
120
110
100
90
University Rating
5
4
3
2
1
SOP
5
4
3
2
1
LOR
5
4
3
2
1
CGPA
10
9
8
7
Research
1
0.5
0
Chance of Admit
1
0.8
0.6
0.4
0
200
400
300
320
340
90
100
110
120
1
2
3
4
5
1
2
3
4
5
1
2
3
4
5
7
8
9
1
Serial No.
GRE Score
TOEFL Score
University Rating
SOP
LOR
CGPA

MINI CHALLENGE #5:

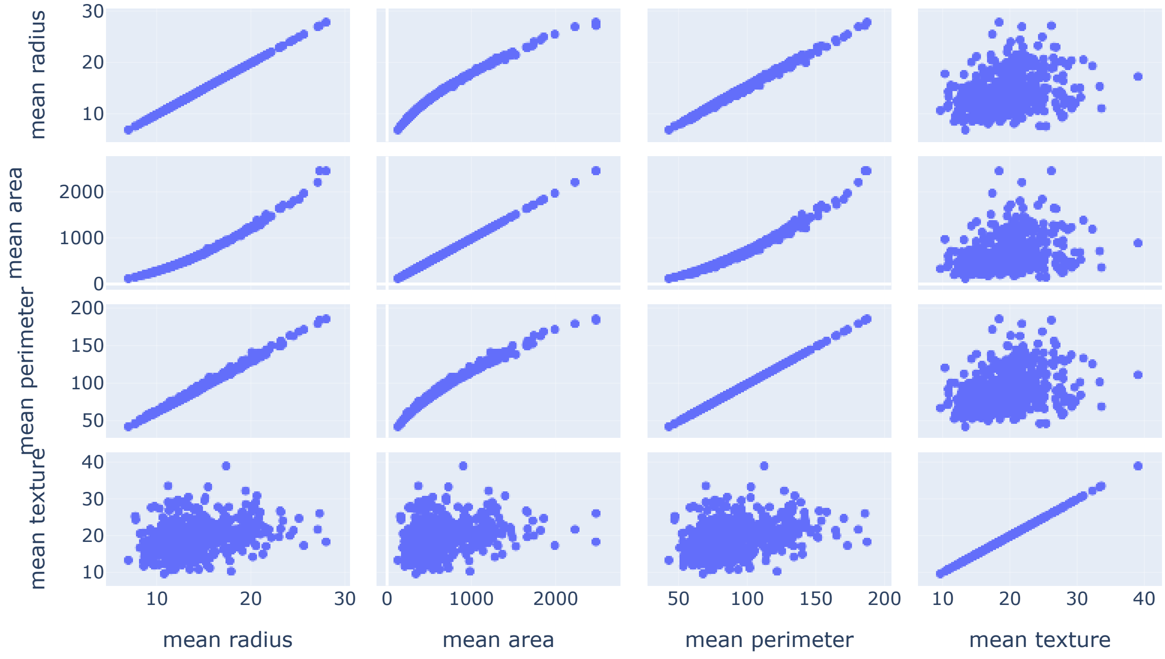
- **Plot the scatter matrix for cancer data, including only the following features: mean radius, mean area, mean perimeter, and mean texture**
- **Plot the scatter matrix for cancer data while color coding the two classes (malignant vs. benign), including only the following features: mean radius, mean area, mean perimeter, and mean texture**
- **What do you infer from this plot**

In [ ]:

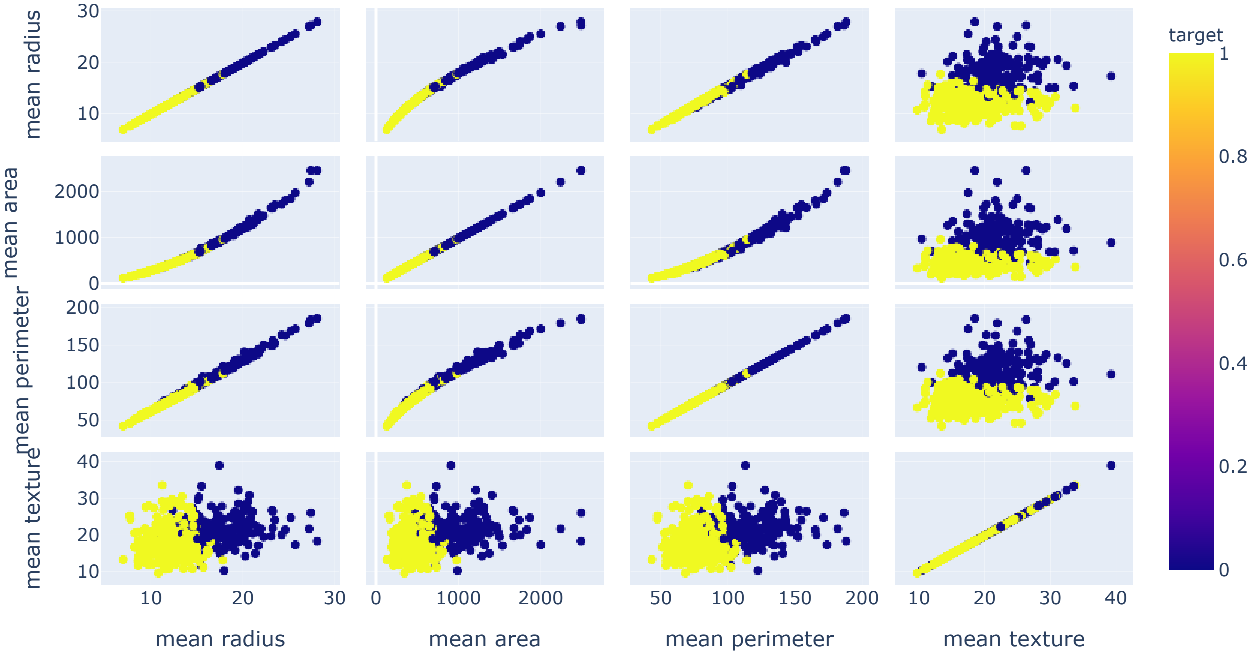
In [ ]:

In [30]:

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



In [33]:
```
fig = px.scatter_matrix(cancer_df, dimensions = ['mean radius', 'mean area', 'mean perimeter', 'me
an texture'], color="target")
fig.show()
```

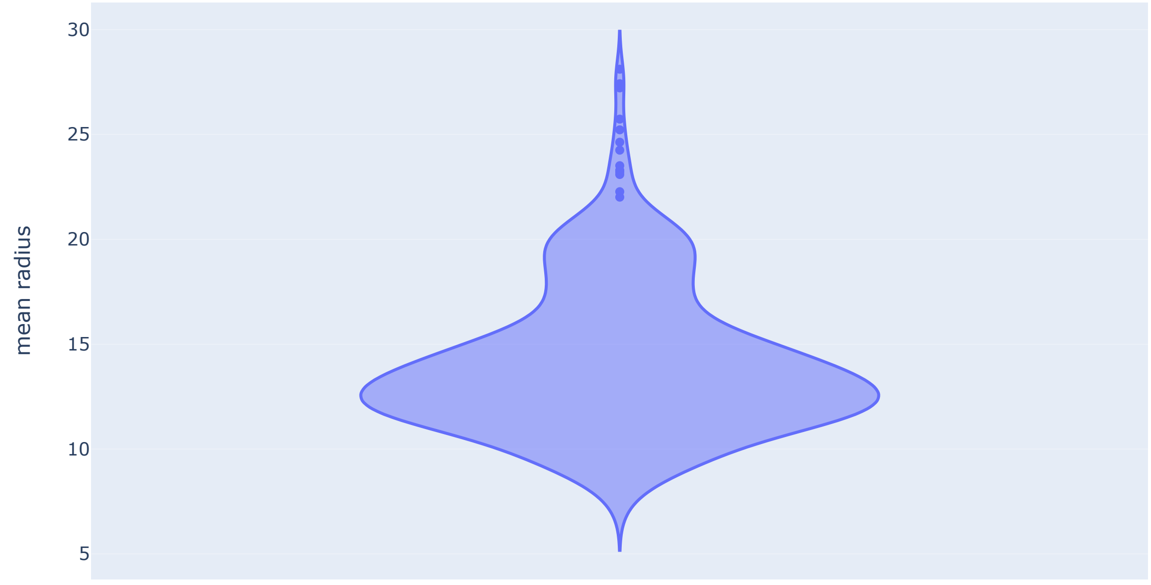


TASK #6: PLOT INTERACTIVE VIOLIN PLOT

In [35]:

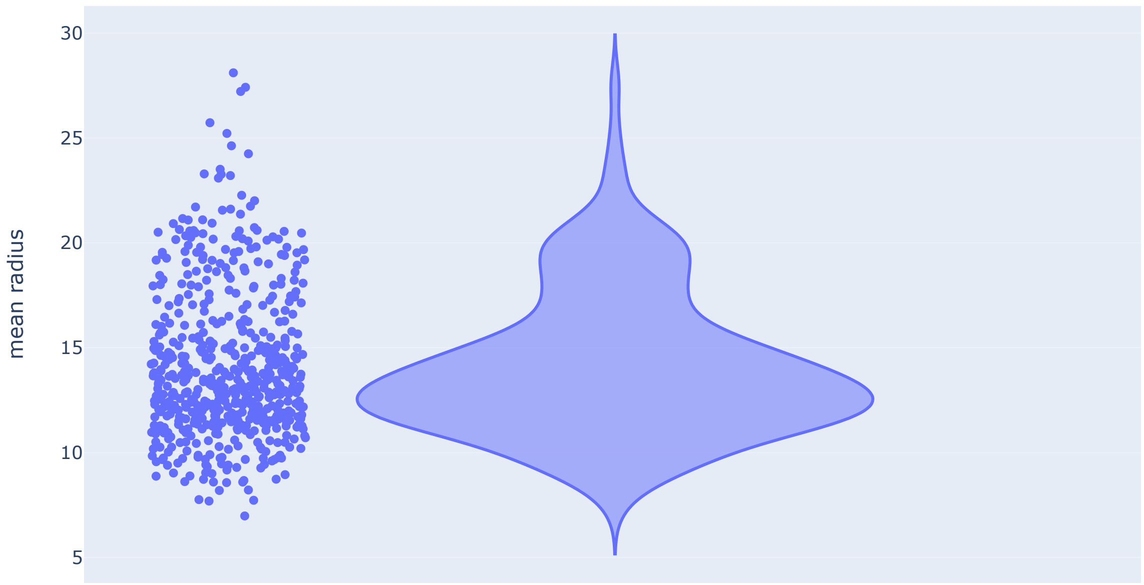
```
# A violin plot is a statistical representation of numerical data.
# It is similar to a box plot, with the addition of a rotated kernel density plot on each side.

fig = px.violin(cancer_df, y = 'mean radius')
fig.show()
```



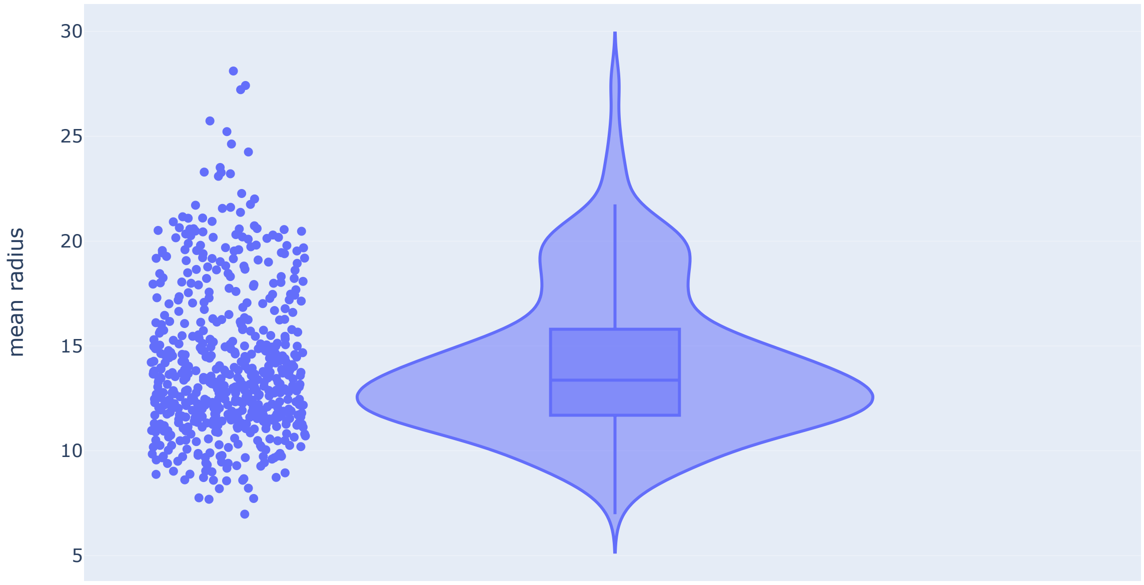
In [36]:

```
# Show data points
fig = px.violin(cancer_df, y = 'mean radius', points = 'all')
fig.show()
```



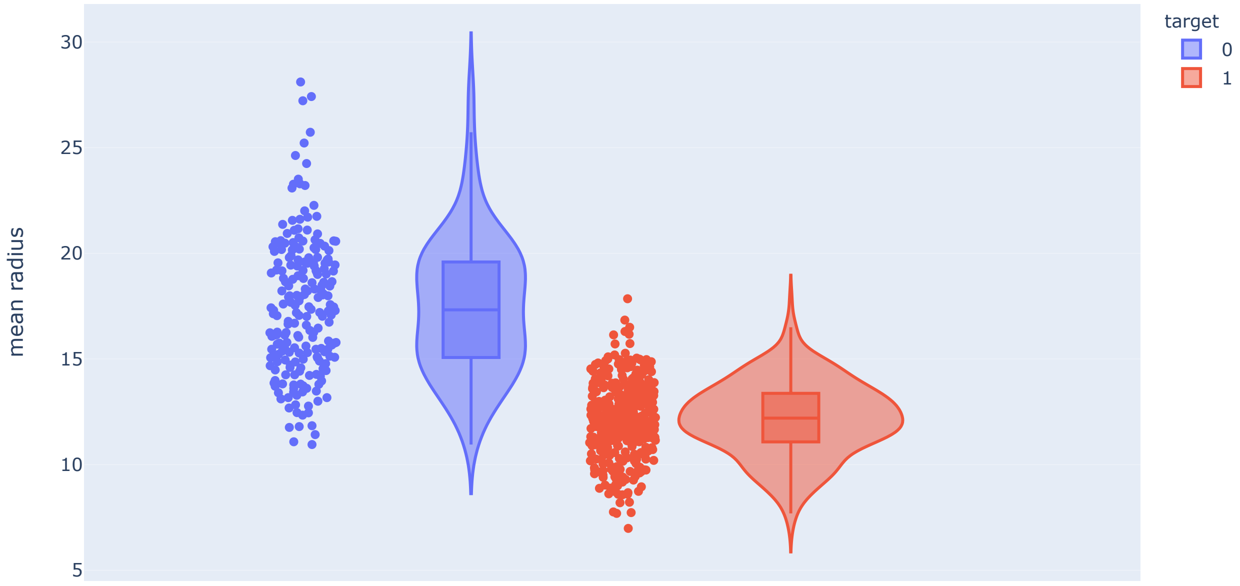
In [37]:

```
# Show data points and box plot
fig = px.violin(cancer_df, y = 'mean radius', box = True, points = 'all')
fig.show()
```



In [38]:

```
# You can also plot multiple violin plots as follows:
fig = px.violin(cancer_df, y = 'mean radius', box = True, points = 'all', color = 'target')
fig.show()
```

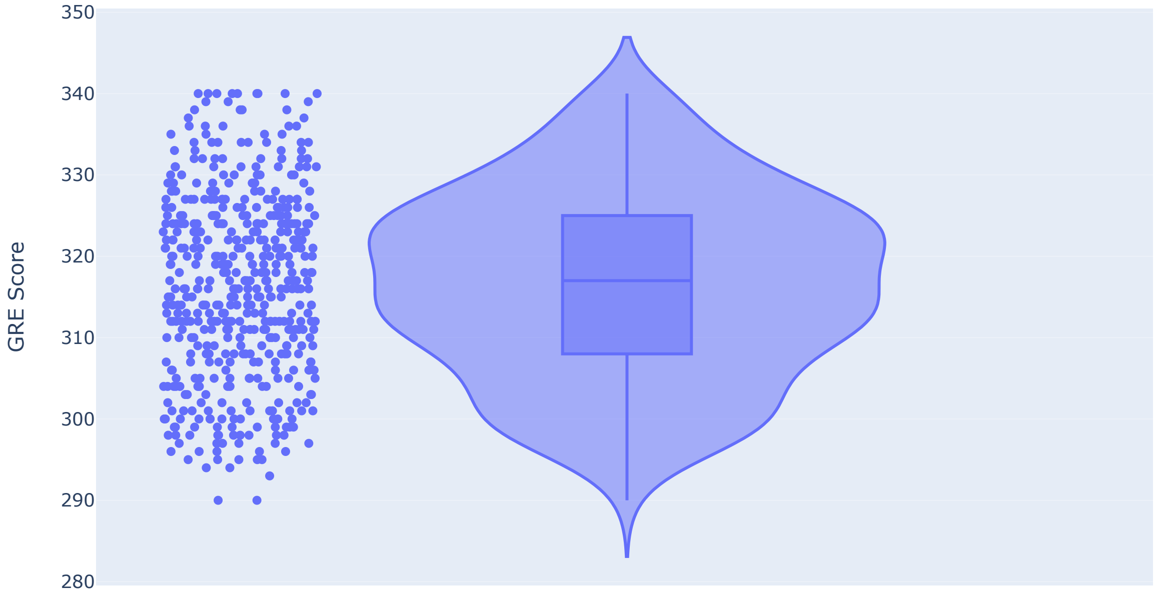


MINI CHALLENGE #6:

- **Plot violin plot for GRE Score in university admission dataset**
- **Using the violin plot, what is the median value of the GRE Score? verify your answer**
- **Calculate the mean value for GRE score and compare it to the median**

In [39]:

```
fig = px.violin(university_df, y = 'GRE Score', box = True, points = 'all')
fig.show()
```



In [40]:

```
university_df.median()
```

```
Serial No.           250.50
GRE Score            317.00
TOEFL Score          107.00
University Rating      3.00
SOP                    3.50
LOR                    3.50
CGPA                   8.56
Research               1.00
Chance of Admit        0.72
dtype: float64
```

In [41]:

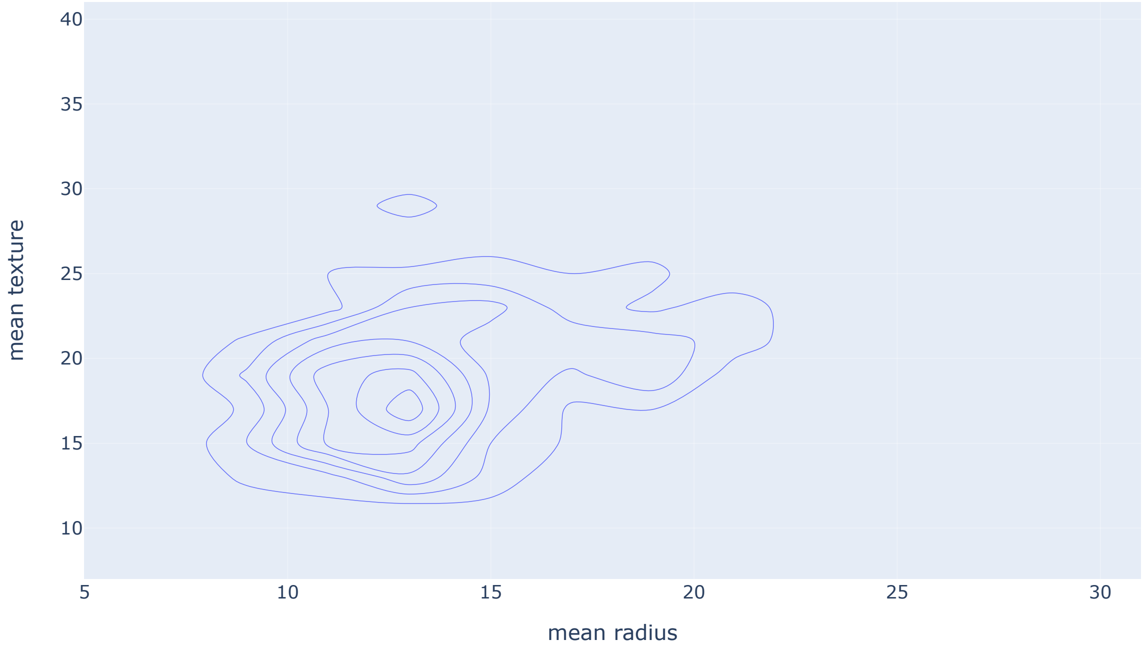
```
university_df.describe()
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
count	500.000000	500.000000	500.000000	500.000000	500.000000	500.00000	500.000000	500.000000	500.00000
mean	250.500000	316.472000	107.192000	3.114000	3.374000	3.48400	8.576440	0.560000	0.72174
std	144.481833	11.295148	6.081868	1.143512	0.991004	0.92545	0.604813	0.496884	0.14114
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.00000	6.800000	0.000000	0.34000
25%	125.750000	308.000000	103.000000	2.000000	2.500000	3.00000	8.127500	0.000000	0.63000
50%	250.500000	317.000000	107.000000	3.000000	3.500000	3.50000	8.560000	1.000000	0.72000
75%	375.250000	325.000000	112.000000	4.000000	4.000000	4.00000	9.040000	1.000000	0.82000
max	500.000000	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	1.000000	0.97000

TASK #7: PLOT INTERACTIVE 2D HISTOGRAM CONTOUR PLOT

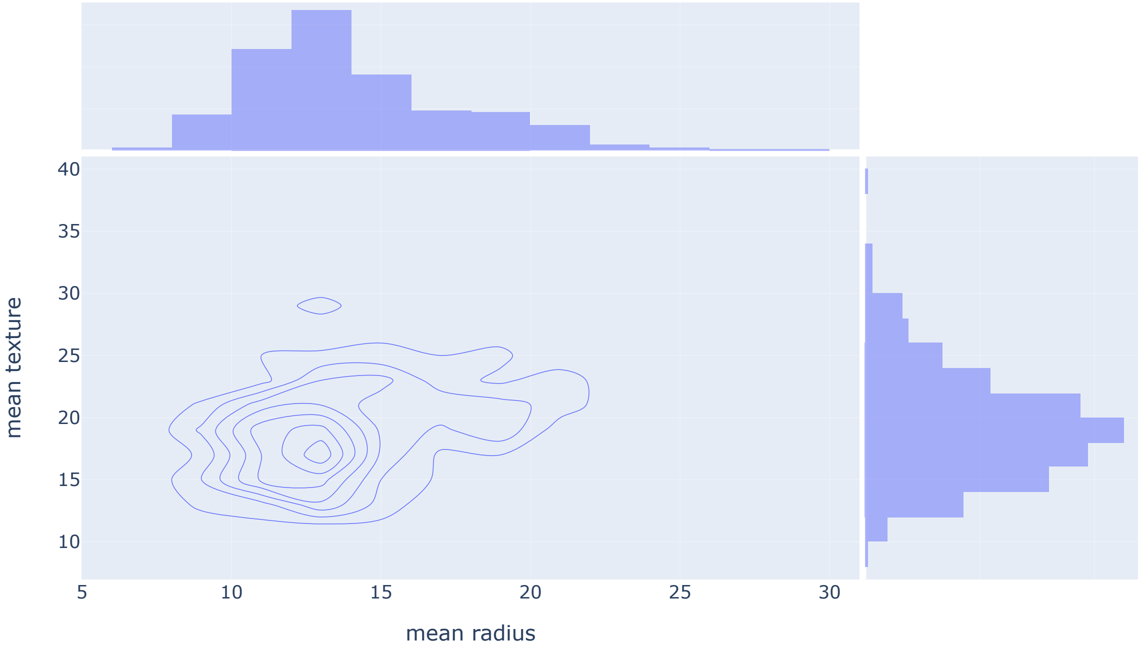
In [42]:

```
# A 2D histogram contour plot, also known as a density contour plot, is a 2-dimensional generalization of a histogram which resembles a contour plot but is computed by grouping a set of points specified by their x and y coordinates into bins,
# and applying an aggregation function such as count or sum (if z is provided) to compute the value to be used to compute contours.
# This kind of visualization (and the related 2D histogram, or density heatmap) is often used to manage over-plotting, or situations where showing large data sets as scatter plots would result in points overlapping each other and hiding patterns.
fig = px.density_contour(cancer_df, x = 'mean radius', y = 'mean texture')
fig.show()
```



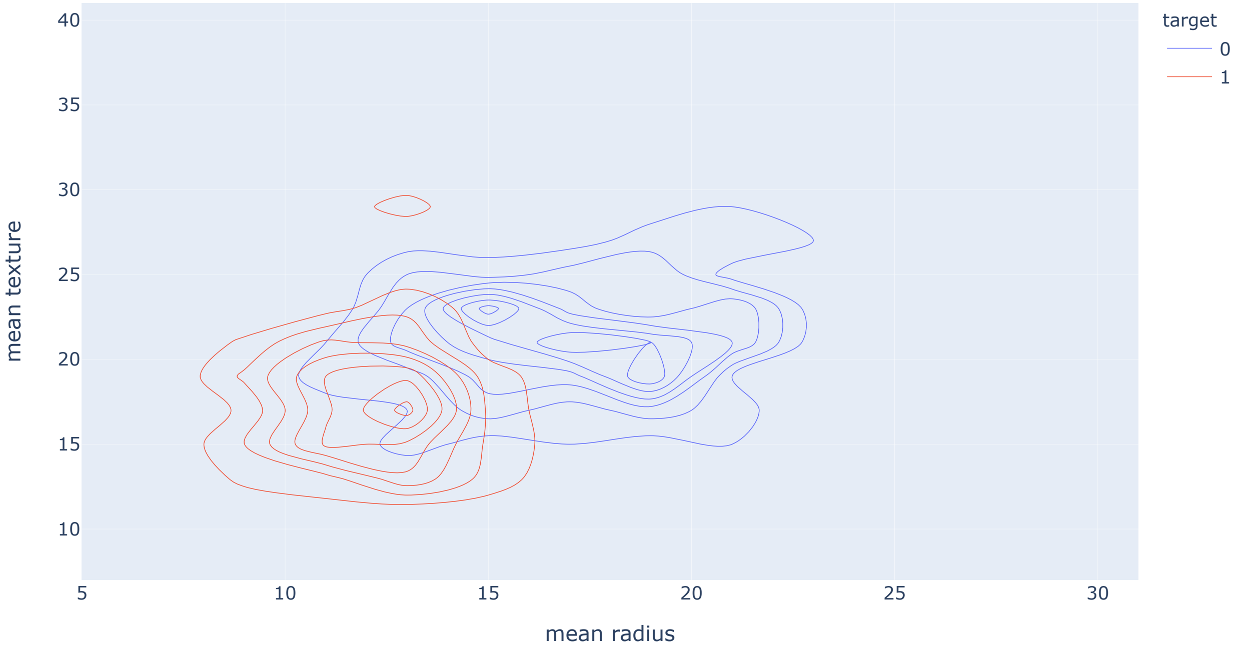
In [43]:

```
# Marginal plots can be added to visualize the 1-dimensional distributions of the two variables.
# Here we use a marginal histogram. Other allowable values are violin, box and rug.
fig = px.density_contour(cancer_df, x = 'mean radius', y = 'mean texture', marginal_x = 'histogram', marginal_y = 'histogram')
fig.show()
```



In [44]:

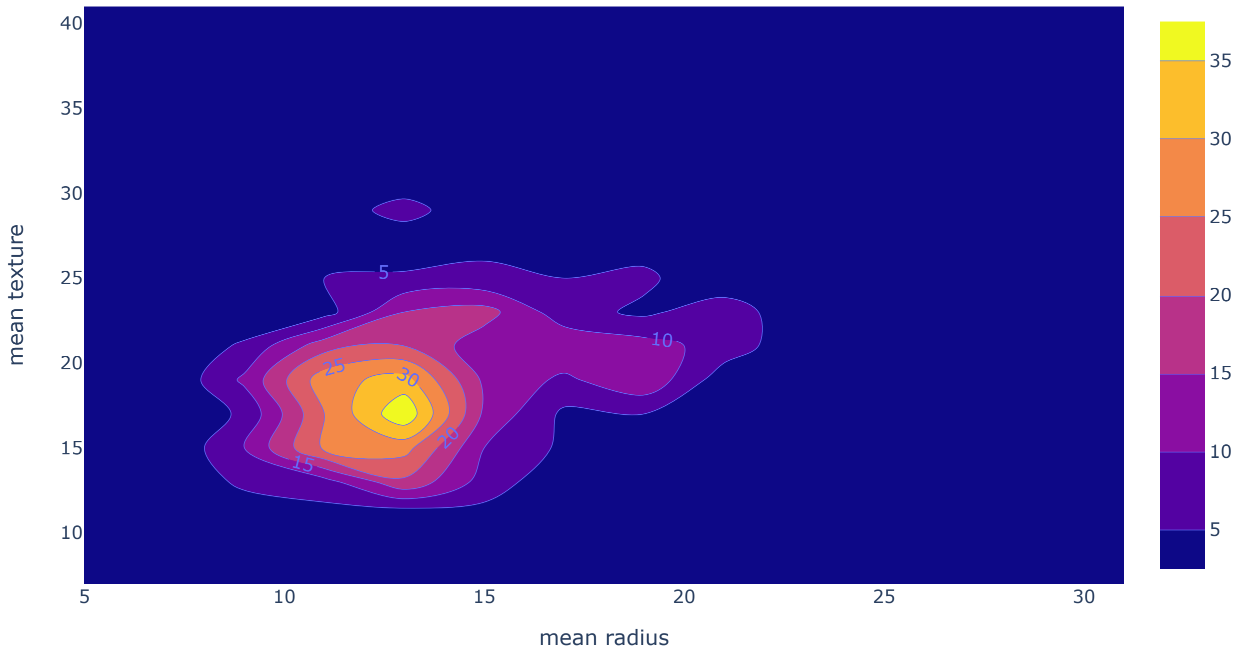
```
fig = px.density_contour(cancer_df, x = 'mean radius', y = 'mean texture', color = 'target')
fig.show()
```



In [46]:

```
# Plotly Express density contours can be continuously-colored and labeled:

fig = px.density_contour(cancer_df, x = 'mean radius', y = 'mean texture')
fig.update_traces(contours_coloring = 'fill', contours_showlabels = True)
```

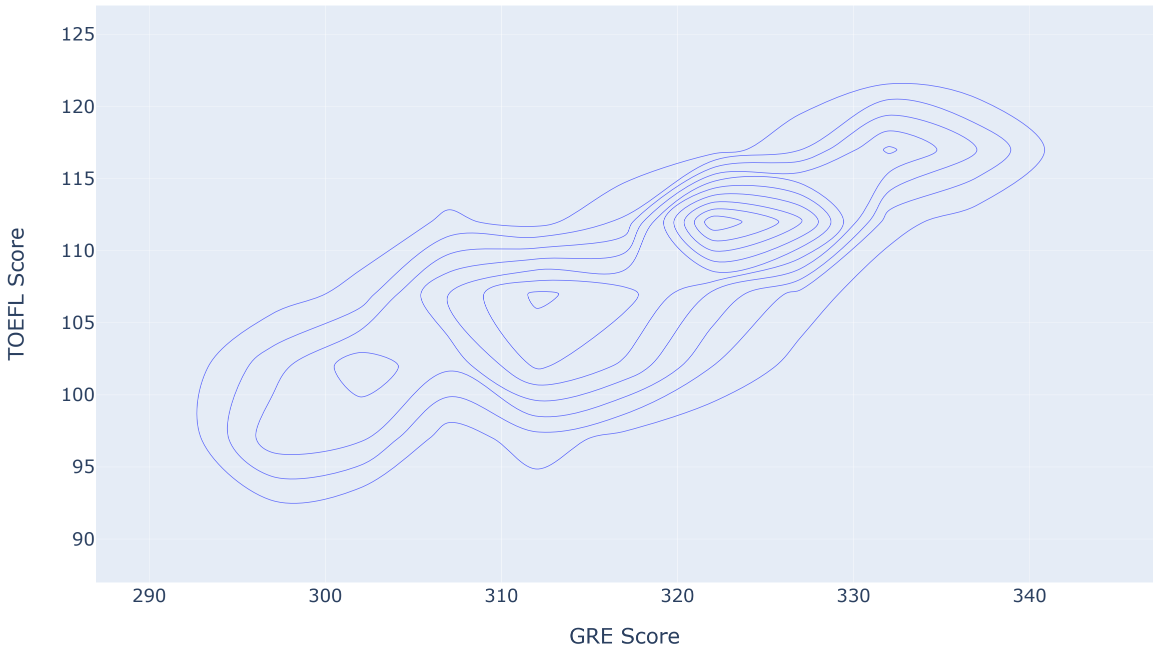


MINI CHALLENGE #7:

- **Plot the density contour plot for the university admission showing GRE Score vs. TOEFL Score**
- **Repeat the plot to show continiously colored data**

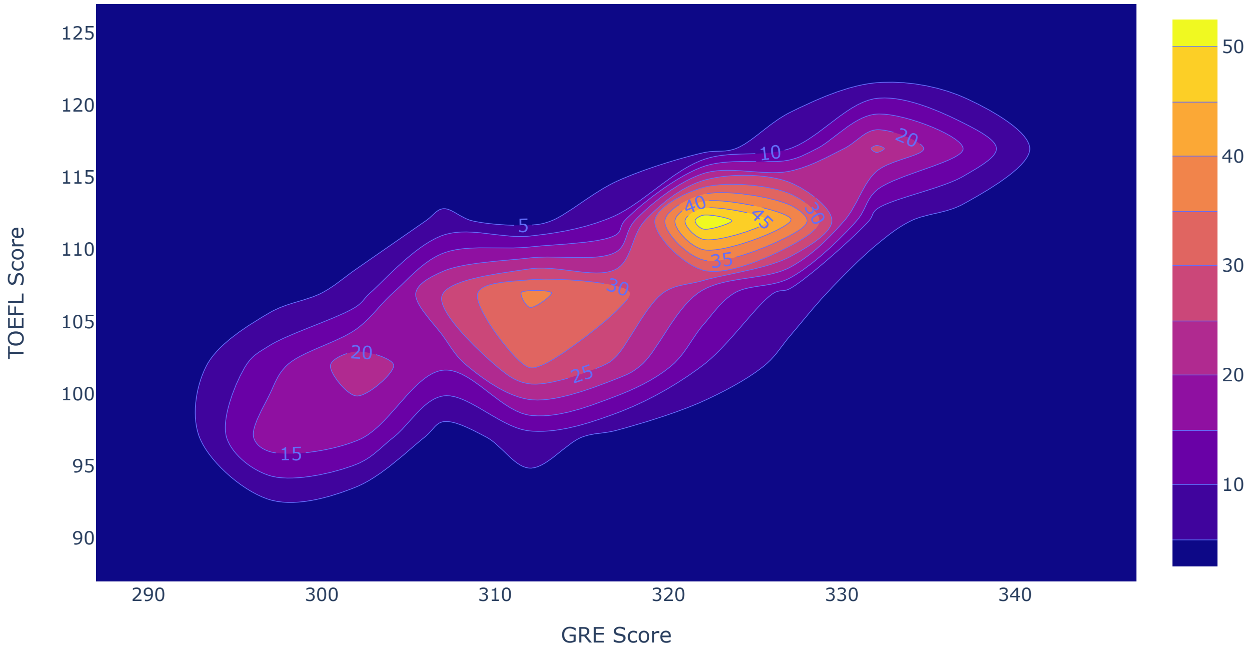
In [48]:

```
fig = px.density_contour(university_df, x = 'GRE Score', y = 'TOEFL Score')
fig.show()
```



In [50]:

```
fig = px.density_contour(university_df, x = 'GRE Score', y = 'TOEFL Score')
fig.update_traces(contours_coloring = 'fill', contours_showlabels = True)
```

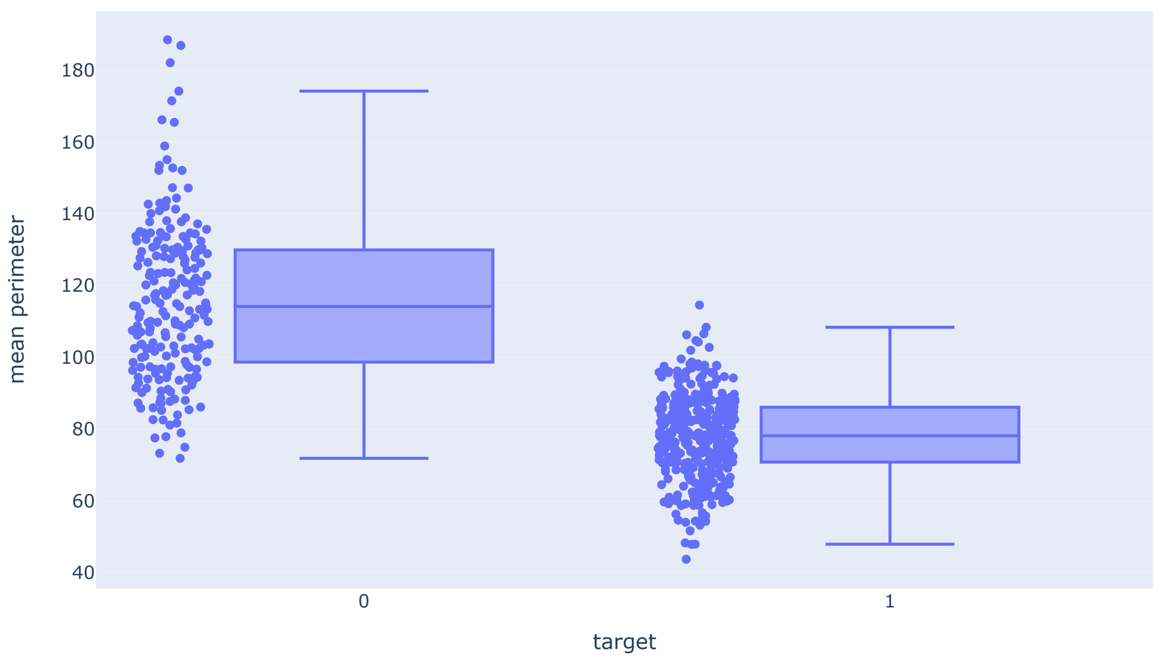


MINI CHALLENGE SOLUTIONS:

MINI CHALLENGE #1 SOLUTION:

- **Plot the boxplot for Mean Perimeter, use points = "all"**

In [23]:
```
fig = px.box(cancer_df, x = "target", y = "mean perimeter", points = "all")
fig.show()
```



MINI CHALLENGE #2 SOLUTION:

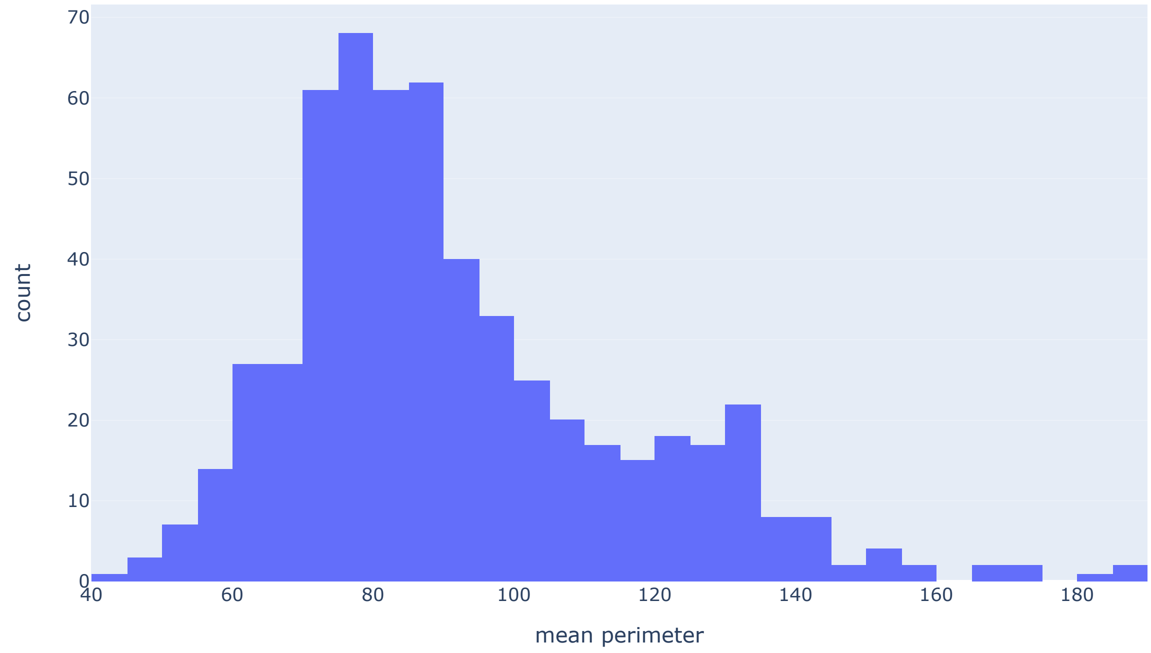
- **Plot the histogram for the mean permieter for the entire dataset**
- **Plot the histogram for the mean perimeter for each of the class independantly**

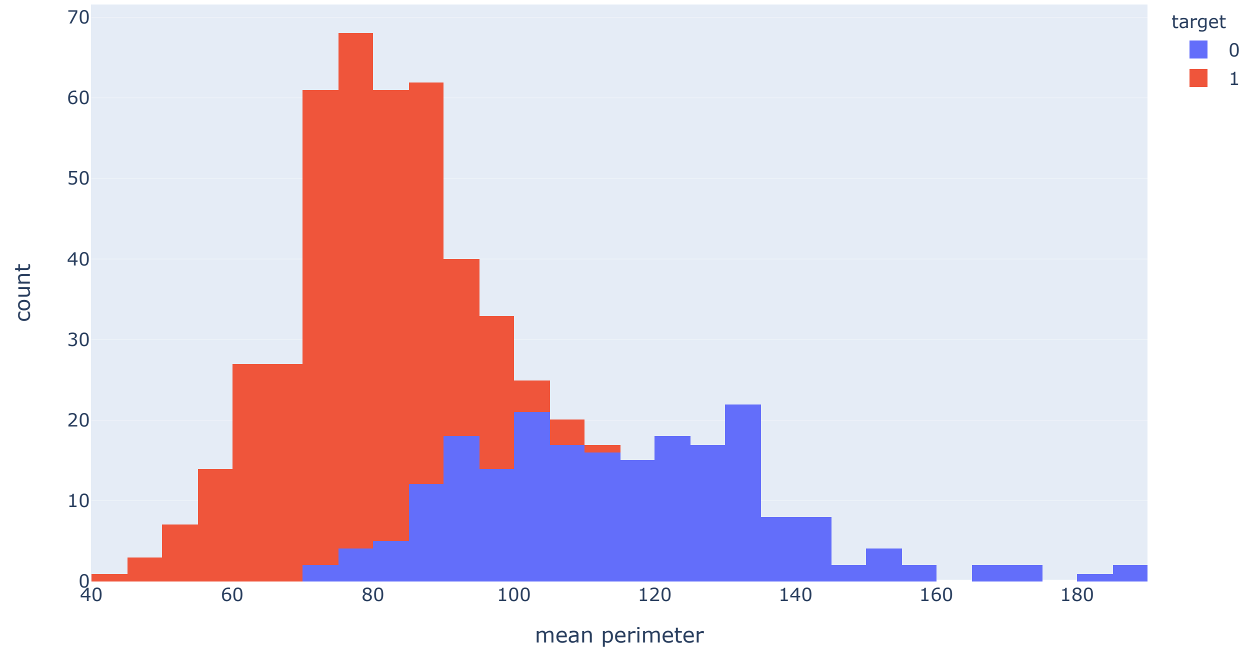
In [24]:

```
# A histogram is representation of the distribution of numerical data, where the data are binned and the count for each bin is represented.

fig = px.histogram(cancer_df, x = "mean perimeter", nbins = 60)
fig.show()

# You can also add the color attribute. This will show the distribution of both classes
fig = px.histogram(cancer_df, x = "mean perimeter", color = 'target', nbins = 60)
fig.show()
```



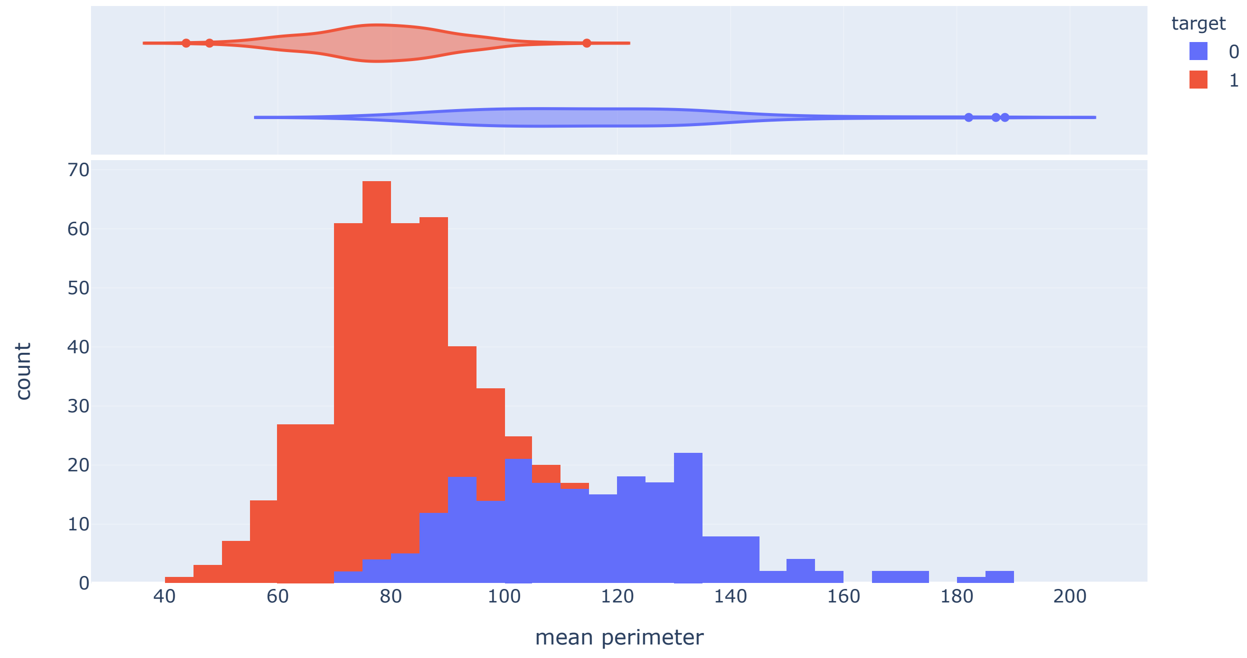


MINI CHALLENGE #3 SOLUTION:

- **Plot the histogram for the mean perimeter using 40 bins and explore a new marginal plot**

In [25]:

```
fig = px.histogram(cancer_df, x = "mean perimeter", color = 'target', marginal="violin", nbins = 40, hover_data = cancer_df.columns)
fig.show()
```

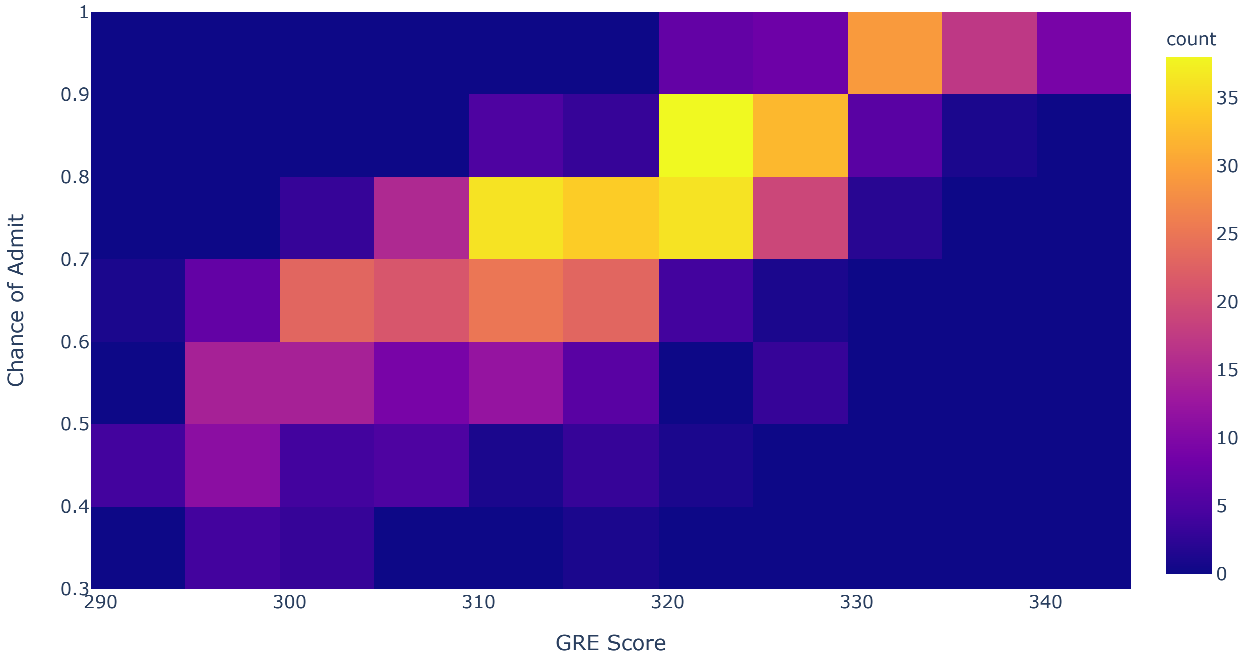


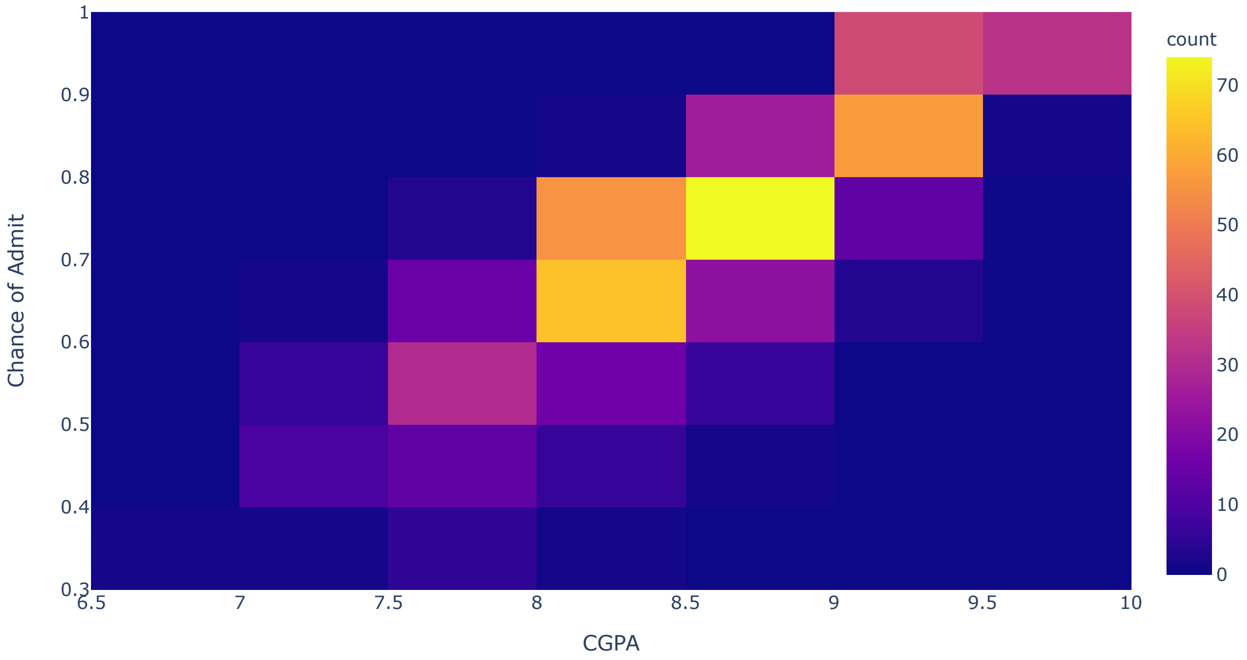
MINI CHALLENGE #4 SOLUTION:

- **Plot density map between GRE Score vs. Chance of admission**
- **Plot density map between GPA vs. Chance of admission**

```
In [26]: fig = px.density_heatmap(university_df, x = "GRE Score", y = "Chance of Admit")
         fig.show()

         fig = px.density_heatmap(university_df, x = "CGPA", y = "Chance of Admit")
         fig.show()
```



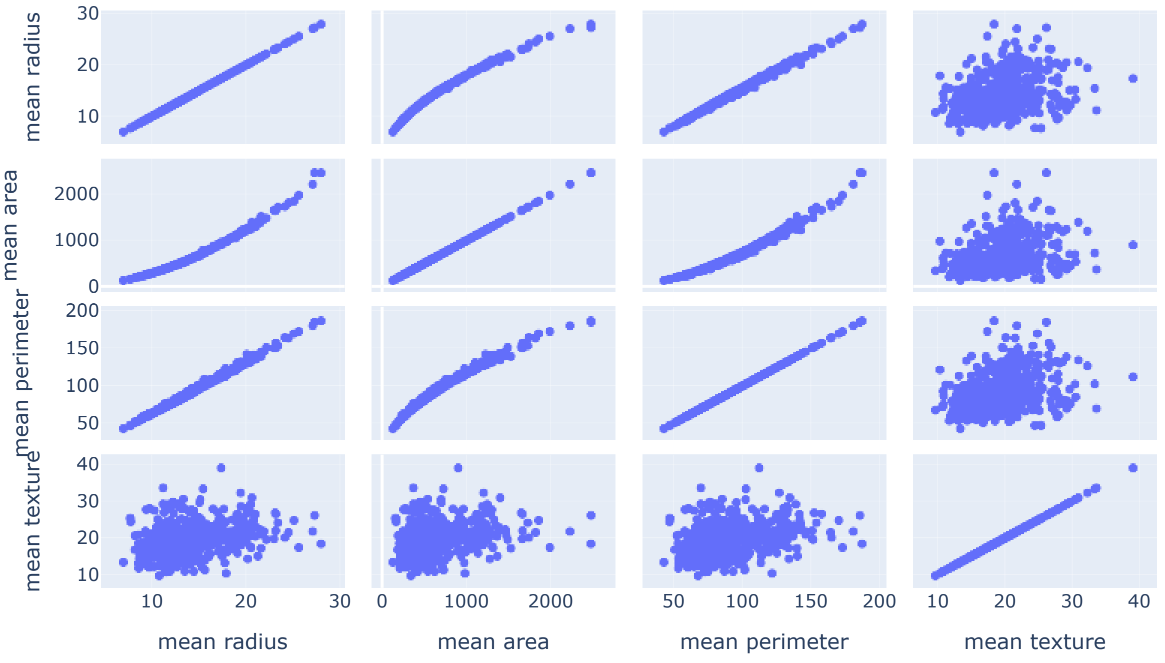


MINI CHALLENGE #5 SOLUTION:

- **Plot the scatter matrix for cancer data, including only the following features: mean radius, mean area, mean perimeter, and mean texture**
- **Plot the scatter matrix for cancer data while color coding the two classes (malignant vs. benign), including only the following features: mean radius, mean area, mean perimeter, and mean texture**
- **What do you infer from this plot**

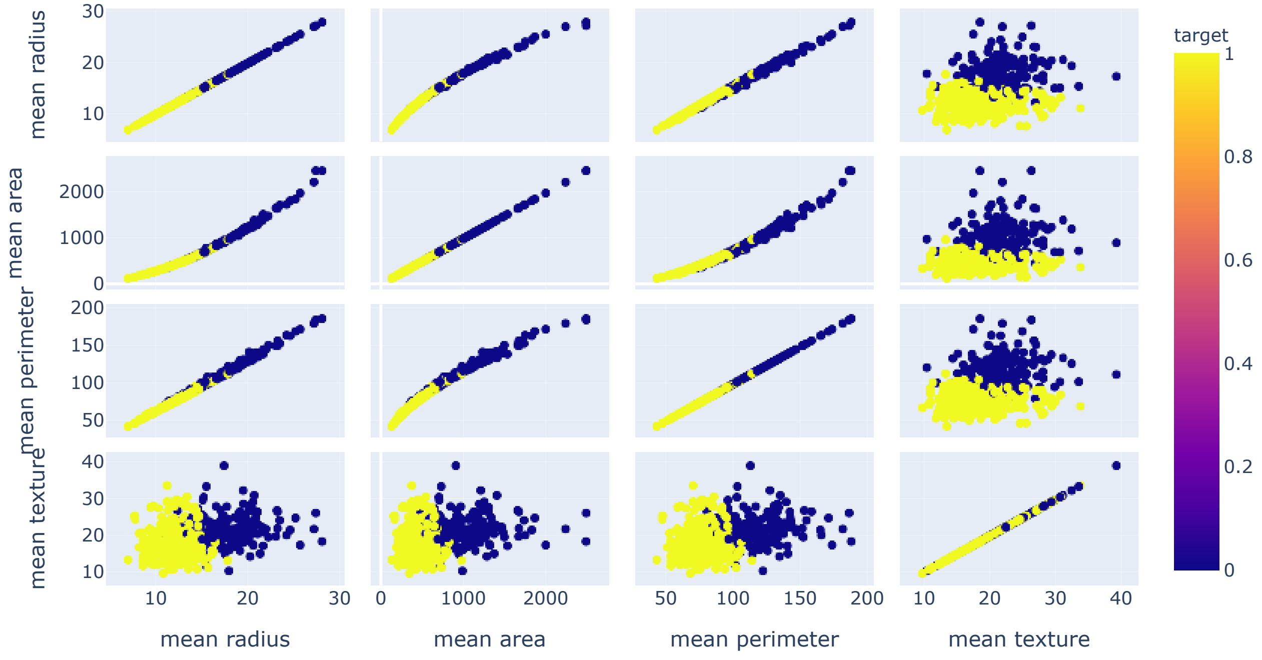
In [27]:

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



In [28]:

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



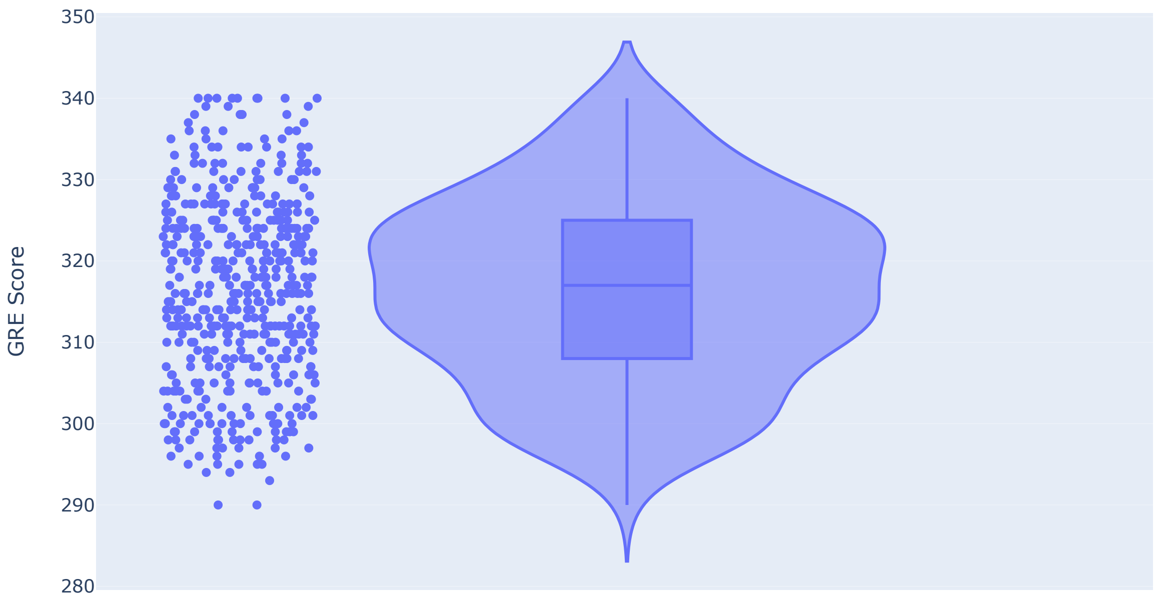
MINI CHALLENGE #6 SOLUTION:

- **Plot violin plot for GRE Score in university admission dataset**
- **Using the violin plot, what is the median value of the GRE Score? verify your answer**
- **Calculate the mean value for GRE score and compare it to the median**

In [29]:

```python
# You can also plot multiple violin plots as follows:
# Median GRE Score = 317
# Mean GRE Score = 316.47
fig = px.violin(university_df, y = "GRE Score", box = True, points = "all", hover_data = university_df.columns)
fig.show()

university_df.median()
university_df.describe()
```



	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
count	500.000000	500.000000	500.000000	500.000000	500.000000	500.00000	500.000000	500.000000	500.00000
mean	250.500000	316.472000	107.192000	3.114000	3.374000	3.48400	8.576440	0.560000	0.72174
std	144.481833	11.295148	6.081868	1.143512	0.991004	0.92545	0.604813	0.496884	0.14114
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.00000	6.800000	0.000000	0.34000
25%	125.750000	308.000000	103.000000	2.000000	2.500000	3.00000	8.127500	0.000000	0.63000
50%	250.500000	317.000000	107.000000	3.000000	3.500000	3.50000	8.560000	1.000000	0.72000
75%	375.250000	325.000000	112.000000	4.000000	4.000000	4.00000	9.040000	1.000000	0.82000
max	500.000000	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	1.000000	0.97000

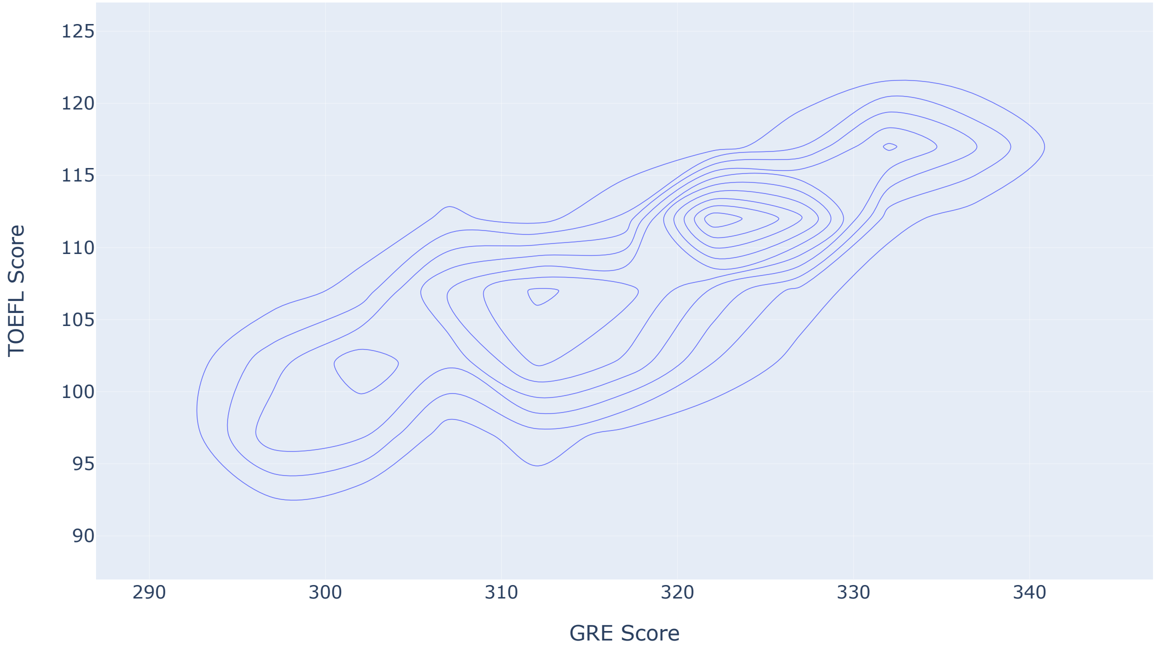
In [ ]:

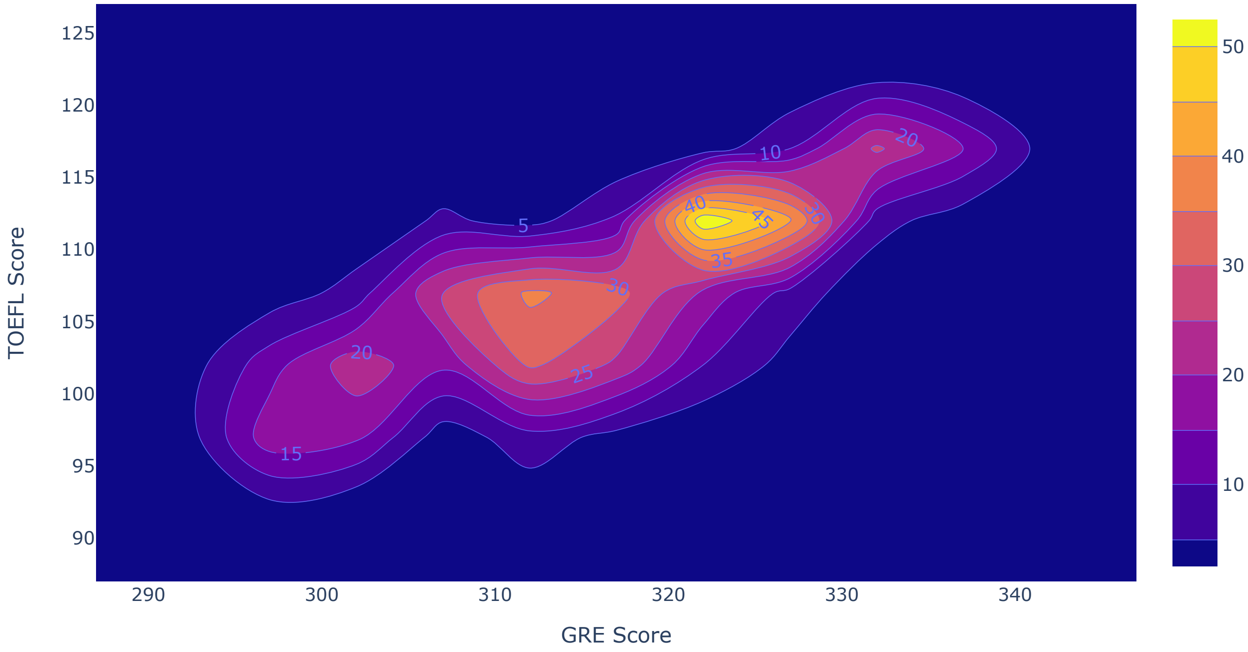
MINI CHALLENGE #7 SOLUTION:

- **Plot the density contour plot for the university admission showing GRE Score vs. TOEFL Score**
- **Repeat the plot to show continiously colored data**

```
In [30]: fig = px.density_contour(university_df, x = "GRE Score", y = "TOEFL Score")
         fig.show()

         fig = px.density_contour(university_df, x = "GRE Score", y = "TOEFL Score")
         fig.update_traces(contours_coloring = "fill", contours_showlabels = True)
```





EXCELLENT JOB!