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


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


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

| | mean radius | mean texture | mean perimeter | mean area | mean smoothness | mean compactness | mean concavity | mean concave points | mean symmetry | mean fractal dimension | ... | worst texture | worst perimeter |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **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 |

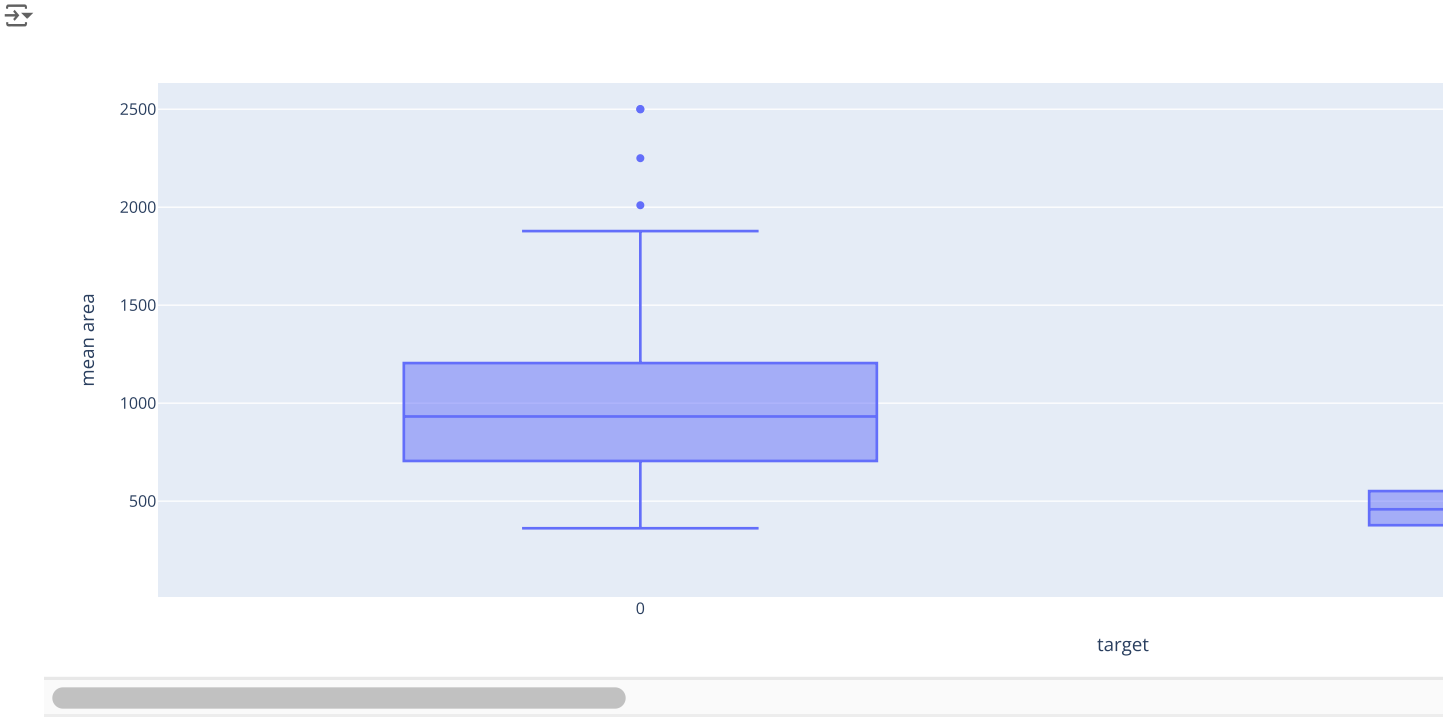
5 rows × 31 columns

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

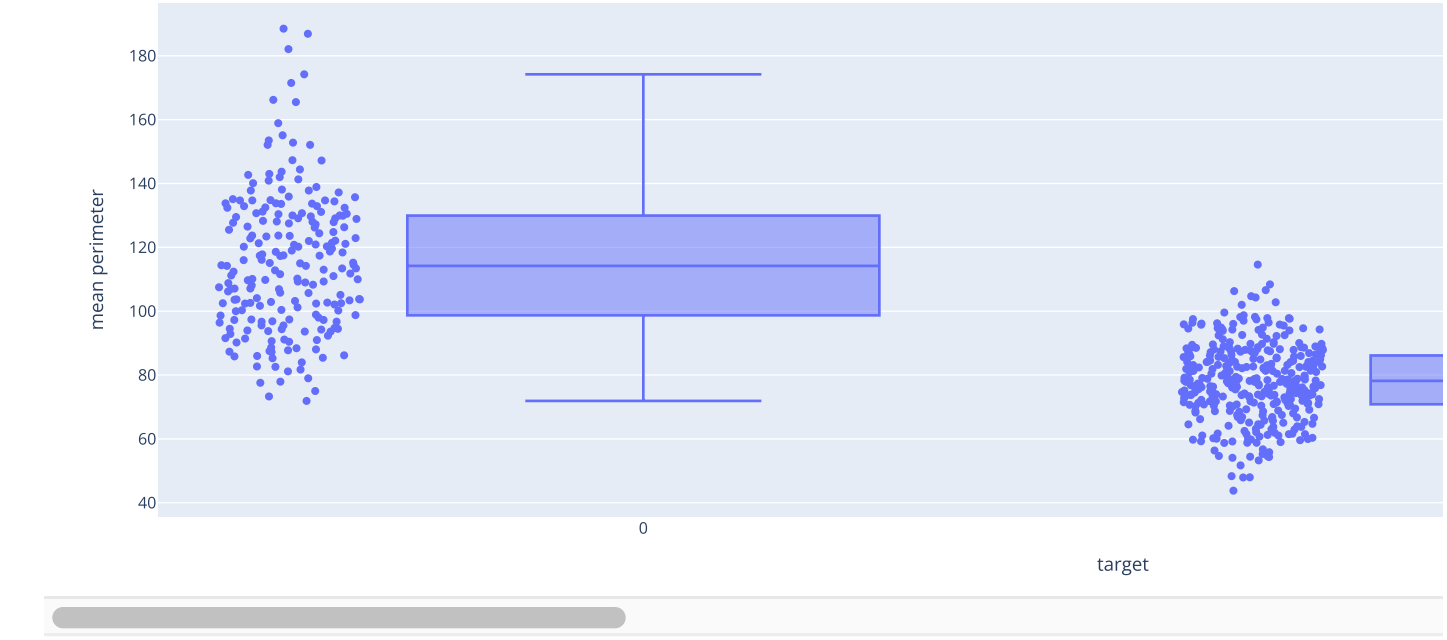
| | mean radius | mean texture | mean perimeter | mean area | mean smoothness | mean compactness | mean concavity | mean concave points | mean symmetry | mean fractal dimension | ... | worst texture | wors perimete |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **564** | 21.56 | 22.39 | 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. |

5 rows × 31 columns

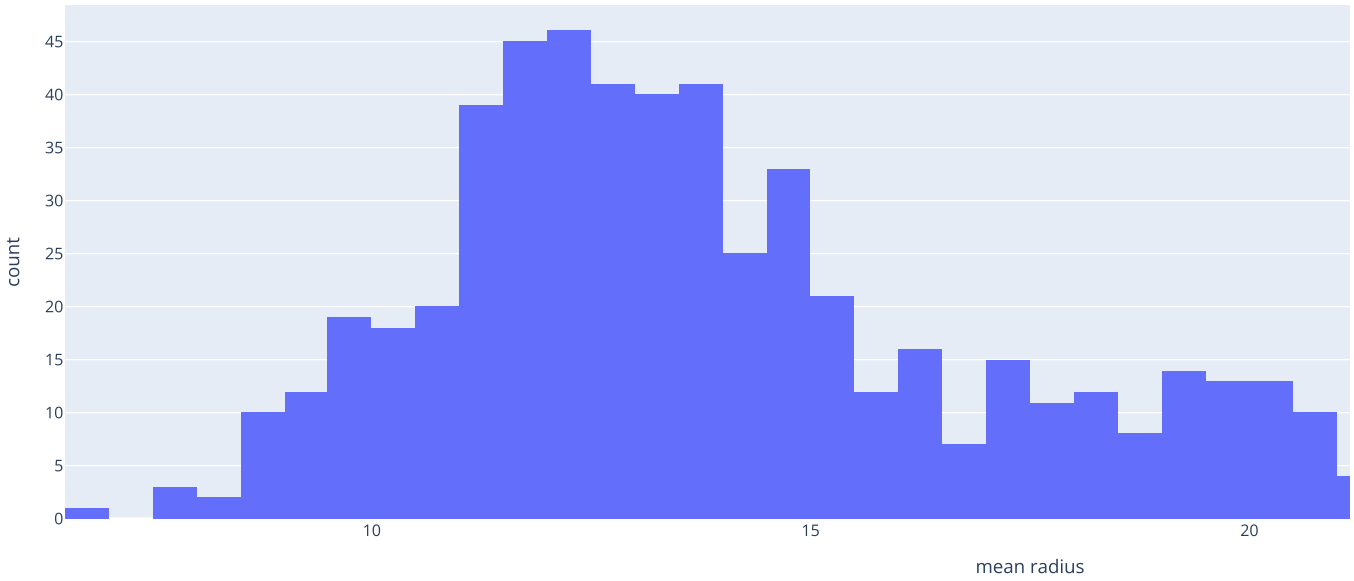
```
#Box plot
fig=px.box(cancer_df, x="target", y="mean area")
fig.show()
```



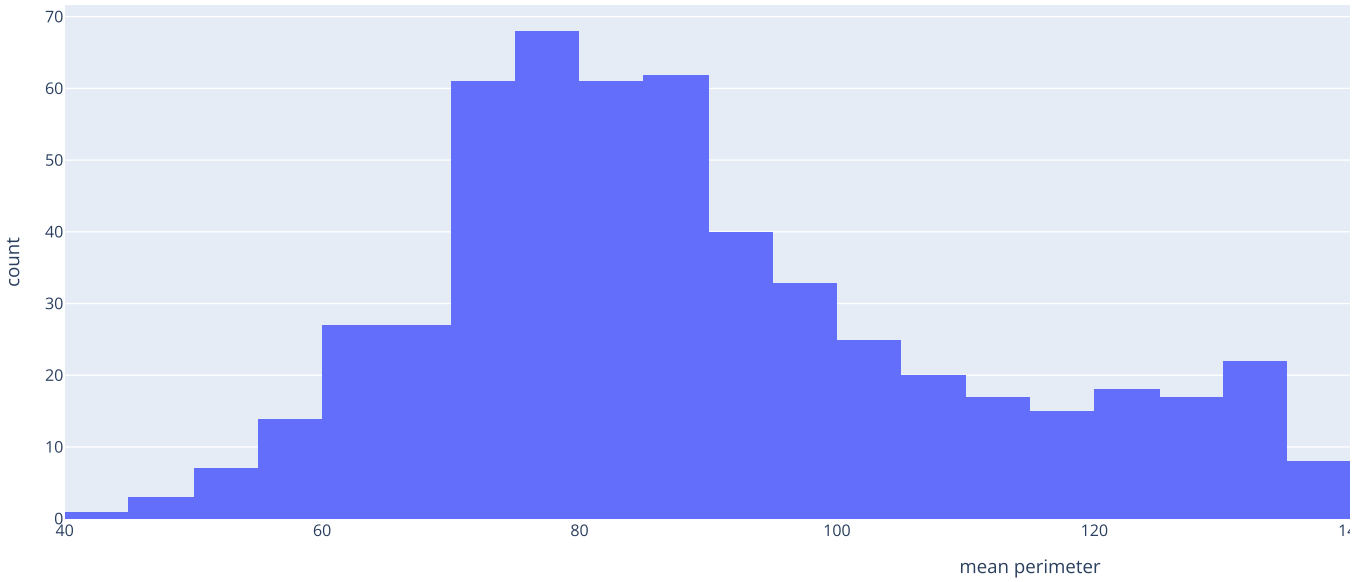

```
#Plot the boxplot for Mean Perimeter, use points = "all"
fig = px.box(cancer_df, x = "target", y = "mean perimeter", points = "all")
fig.show()
```



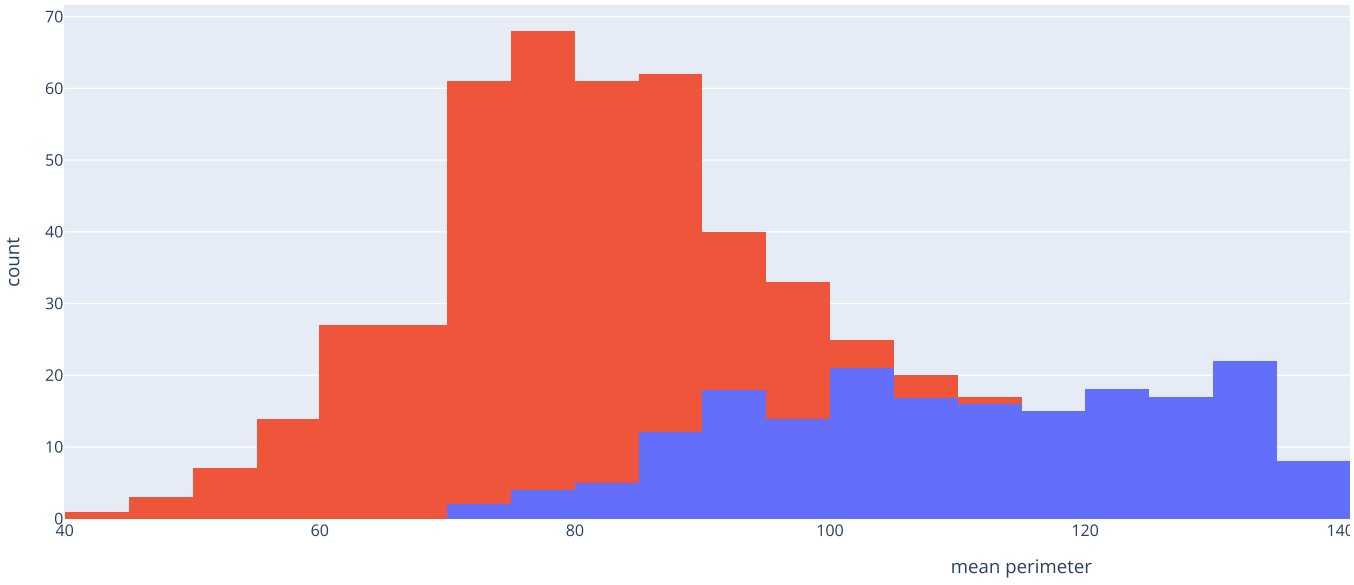

```
 #A histogram is representation of the distribution of numerical data, where the data are binned
 fig = px.histogram(cancer_df, x = "mean radius", nbins = 60)
fig.show()
```



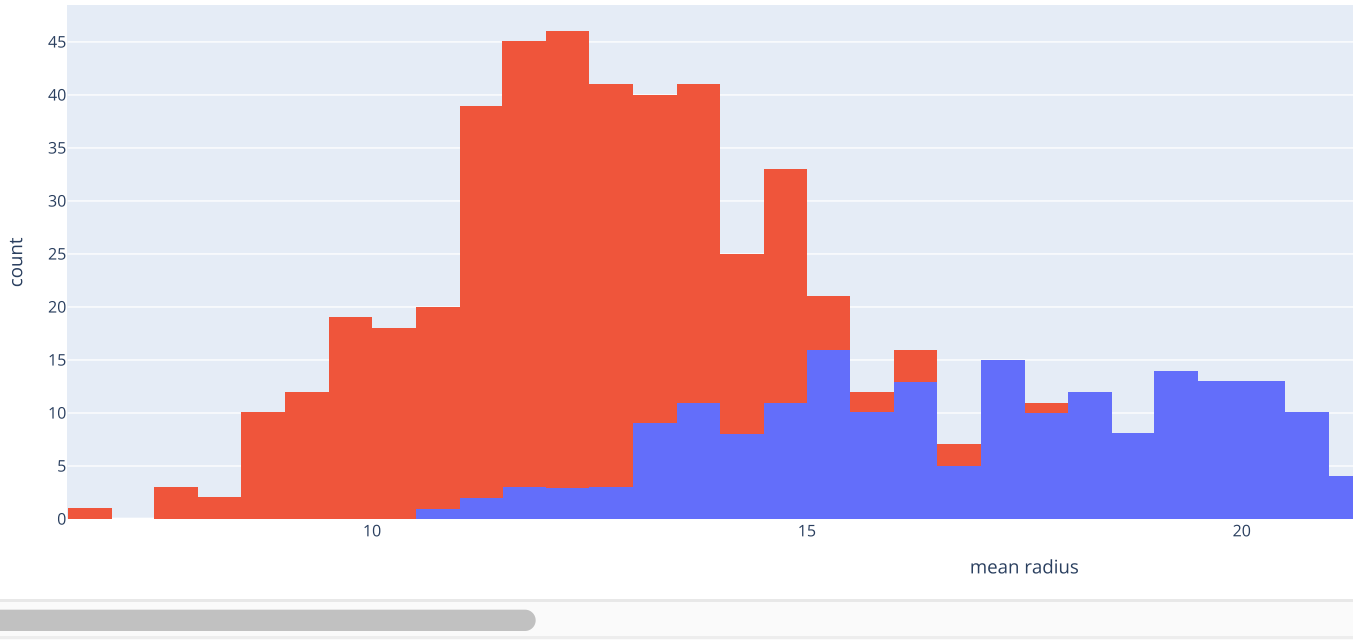

```
#Plot the histogram for the mean permieter for the entire dataset

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



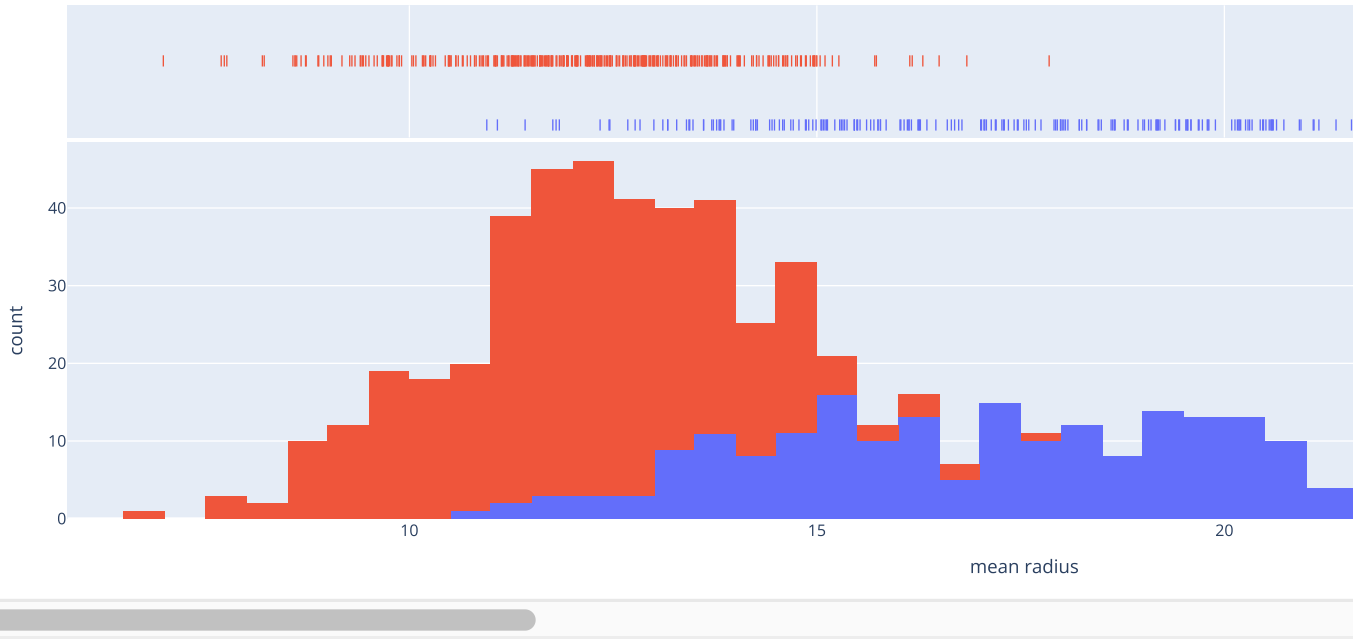

```
#Plot the histogram for the mean perimeter for each of the class independantly
fig = px.histogram(cancer_df, x = "mean perimeter", color = 'target', nbins = 60)
fig.show()
```



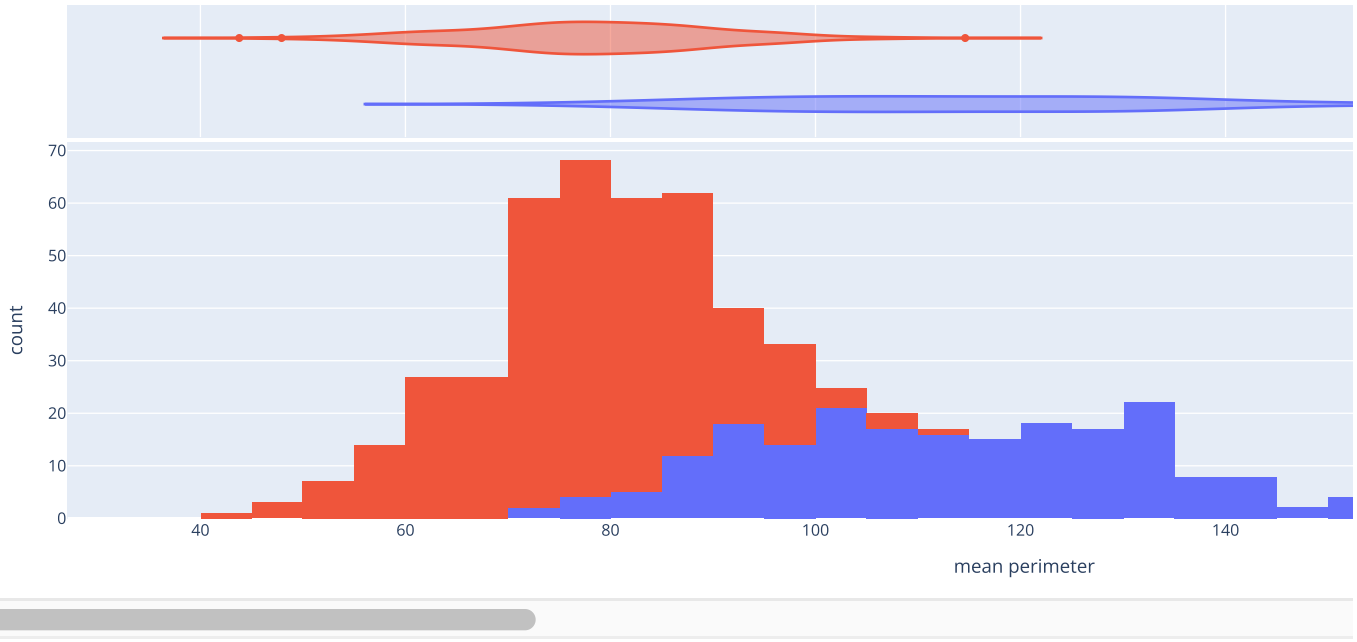

```
fig = px.histogram(cancer_df, x = "mean radius", color = 'target', nbins = 60)
fig.show()
```



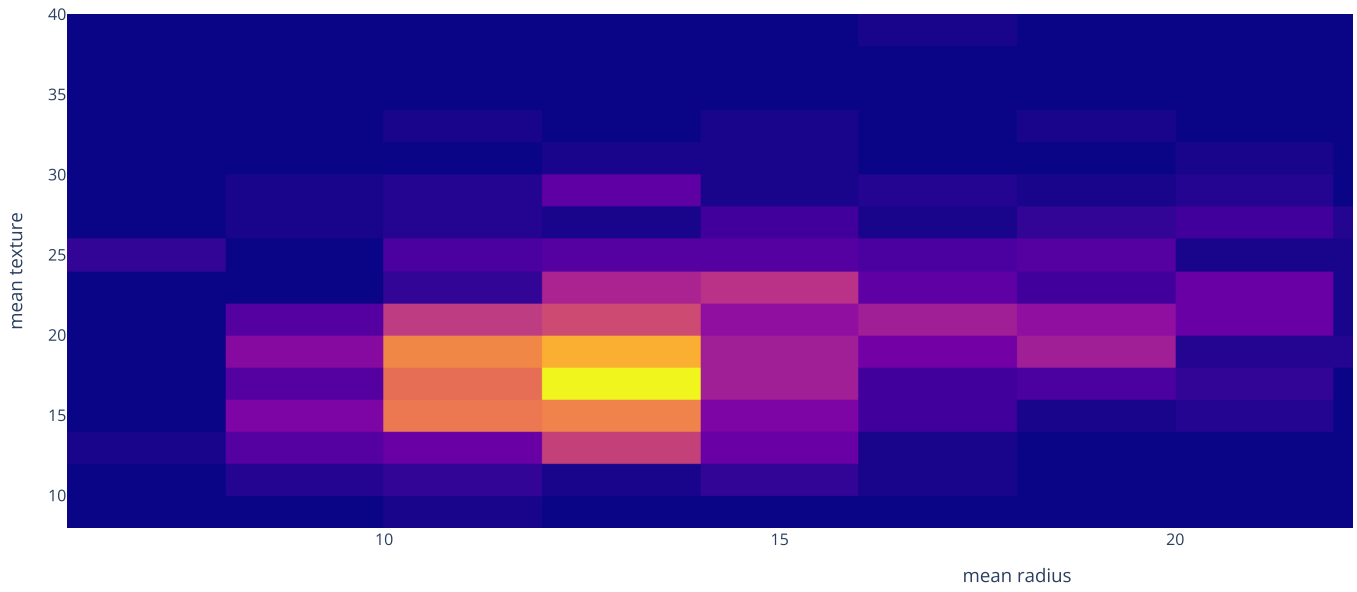

```
# With the marginal keyword, a subplot is drawn alongside the histogram, visualizing the distribution.
fig = px.histogram(cancer_df, x = "mean radius", color = 'target', marginal = 'rug', nbins = 60)
fig.show()
```



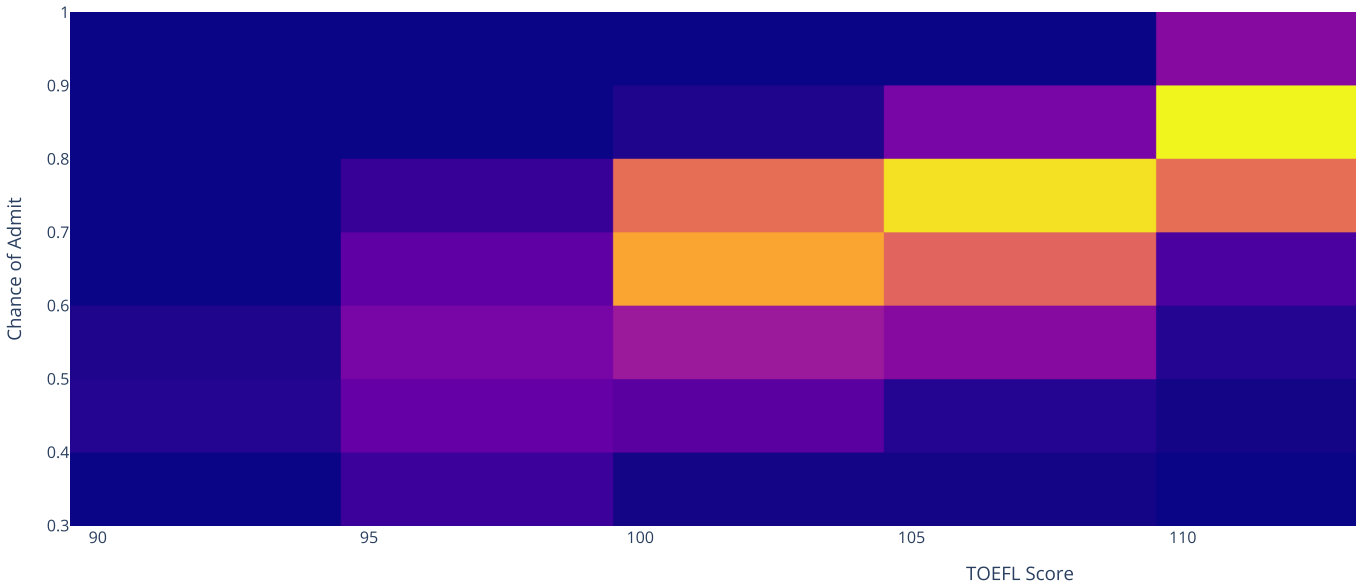

```
#Plot the histogram for the mean perimeter using 40 bins and explore a new marginal plot
fig = px.histogram(cancer_df, x = "mean perimeter", color = 'target', marginal="violin", nbins = 40, hover_data = cancer_df.colu
fig.show()
```



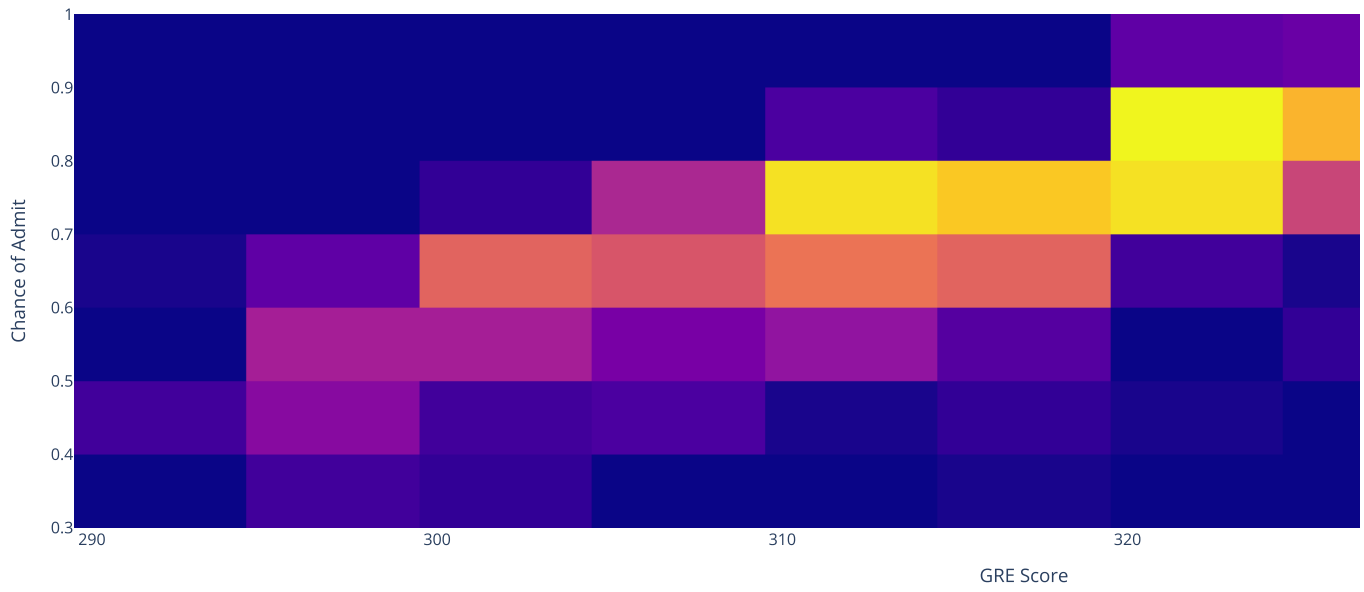

```
# A 2D histogram, also known as a density heatmap, is the 2-dimensional generalization of a histogram which resembles a heatmap
# and applying an aggregation function such as count or sum (if z is provided) to compute the color of the tile representing the
# This kind of visualization (and the related 2D histogram contour, or density contour) is often used to manage over-plotting, c
fig=px.density_heatmap(cancer_df, x="mean radius", y="mean texture")
fig.show()
```



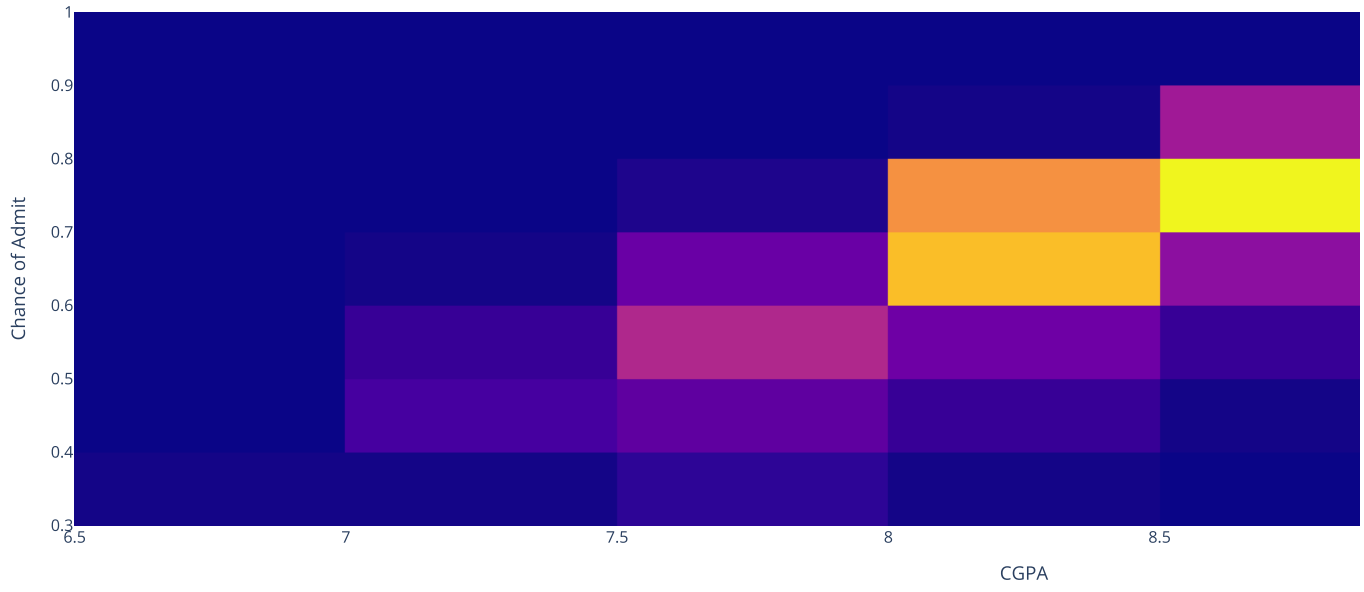

```
# read univeristy_admission.csv dataset
university_df=pd.read_csv("/content/university_admission.csv")


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



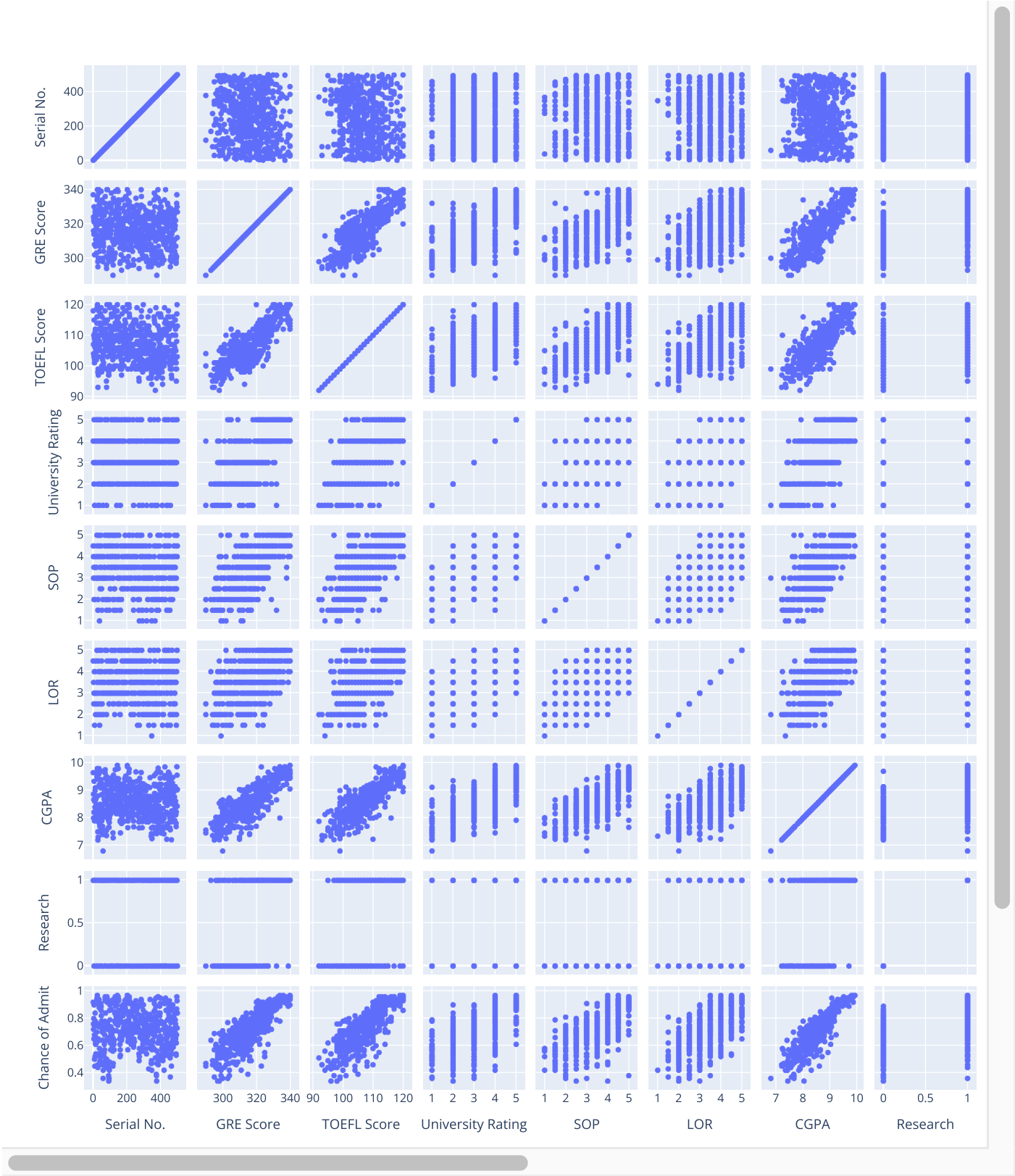

```
#Plot density map between GRE Score vs. Chance of admission
fig = px.density_heatmap(university_df, x = "GRE Score", y = "Chance of Admit")
fig.show()
```



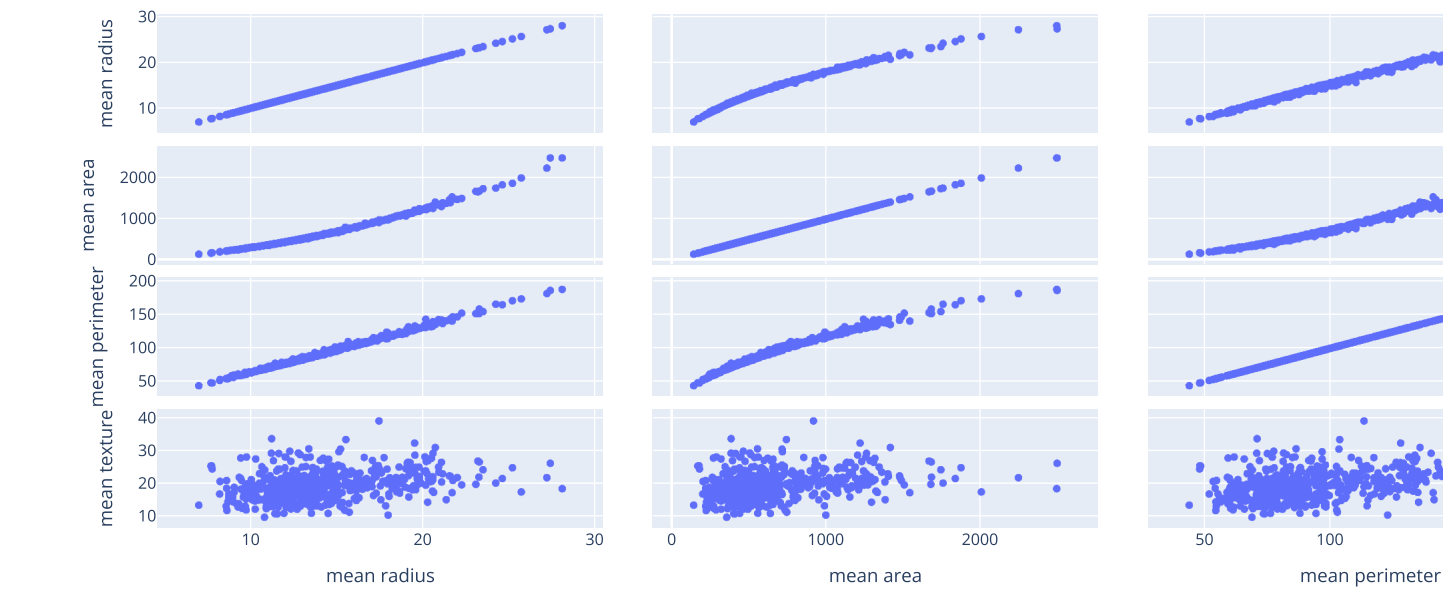

```
#Plot density map between GPA vs. Chance of admission
fig = px.density_heatmap(university_df, x = "CGPA", y = "Chance of Admit")
fig.show()
```



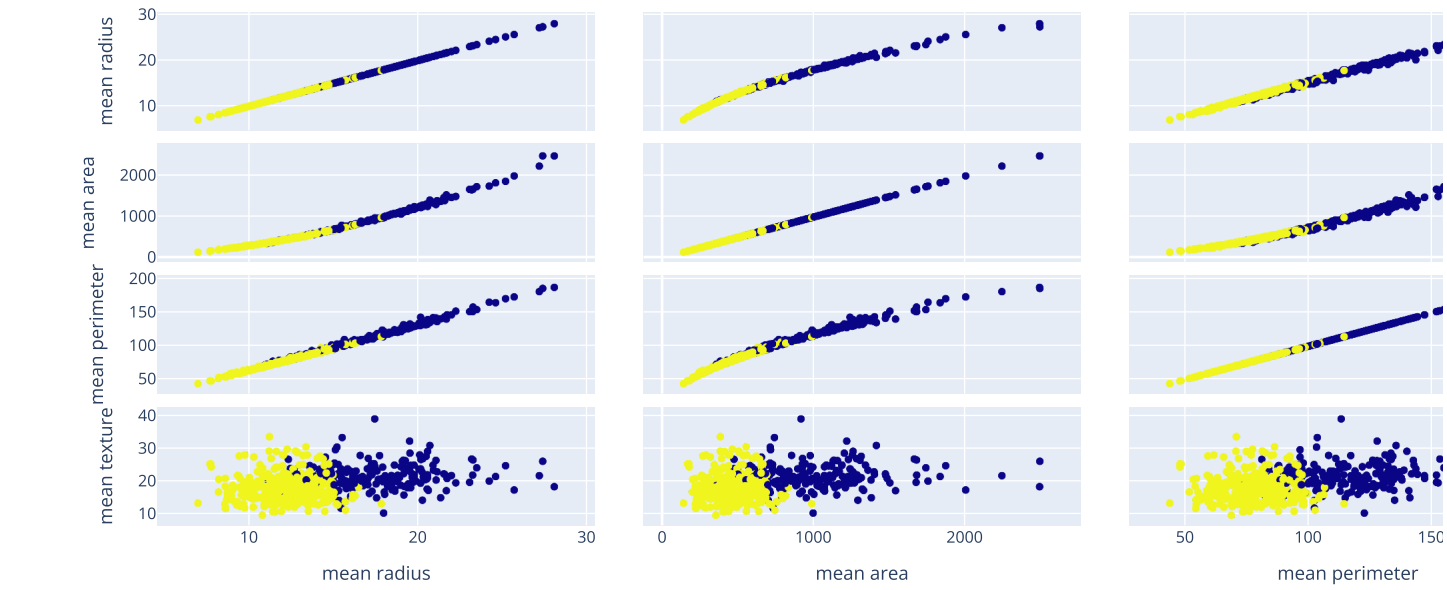

```
# A scatterplot matrix is a matrix associated to n numerical arrays (data variables), $X_1,X_2,…,X_n$ , of the same length.
fig = px.scatter_matrix(university_df, width=1200, height=1200)
fig.show()
```



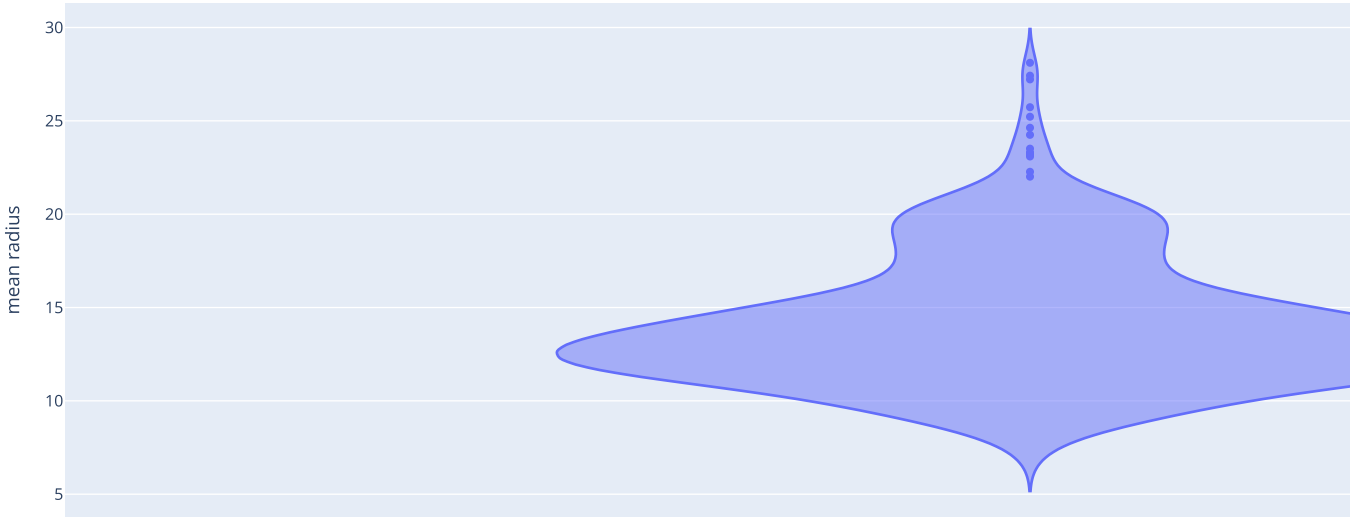

```
#Plot the scatter matrix for cancer data, including only the following features: mean radius, mean area, mean perimeter, and mea
fig = px.scatter_matrix(cancer_df, dimensions = ['mean radius', 'mean area', 'mean perimeter', 'mean texture'])
fig.show()
```



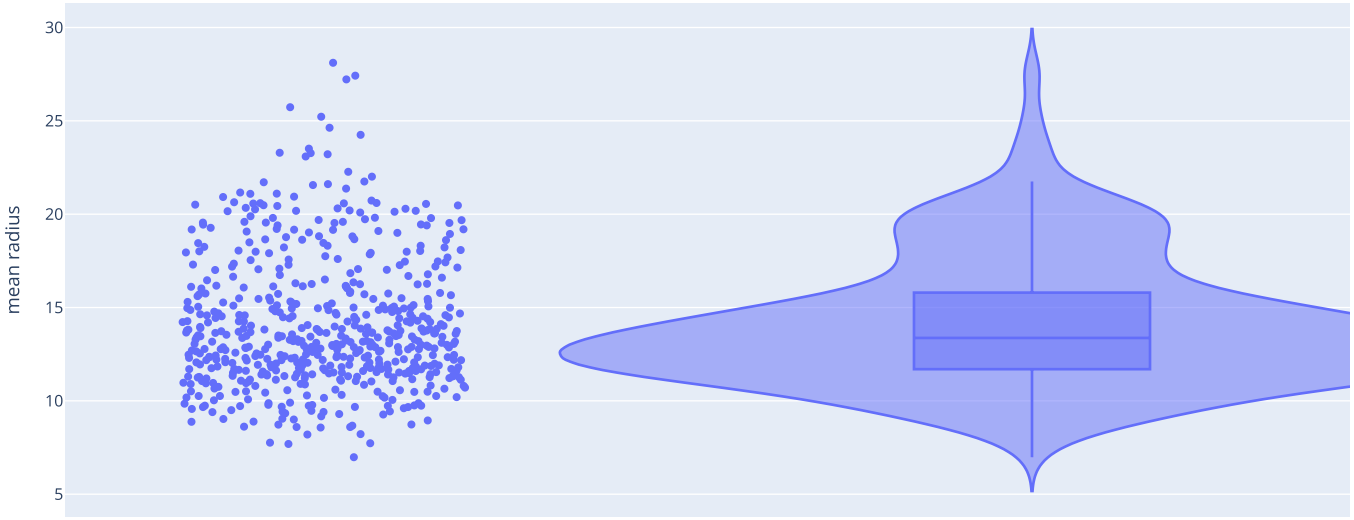

```
#Plot the scatter matrix for cancer data while color coding the two classes (malignant vs. benign), including only the following
fig = px.scatter_matrix(cancer_df, dimensions = ['mean radius', 'mean area', 'mean perimeter', 'mean texture'], color="target")
fig.show()
```



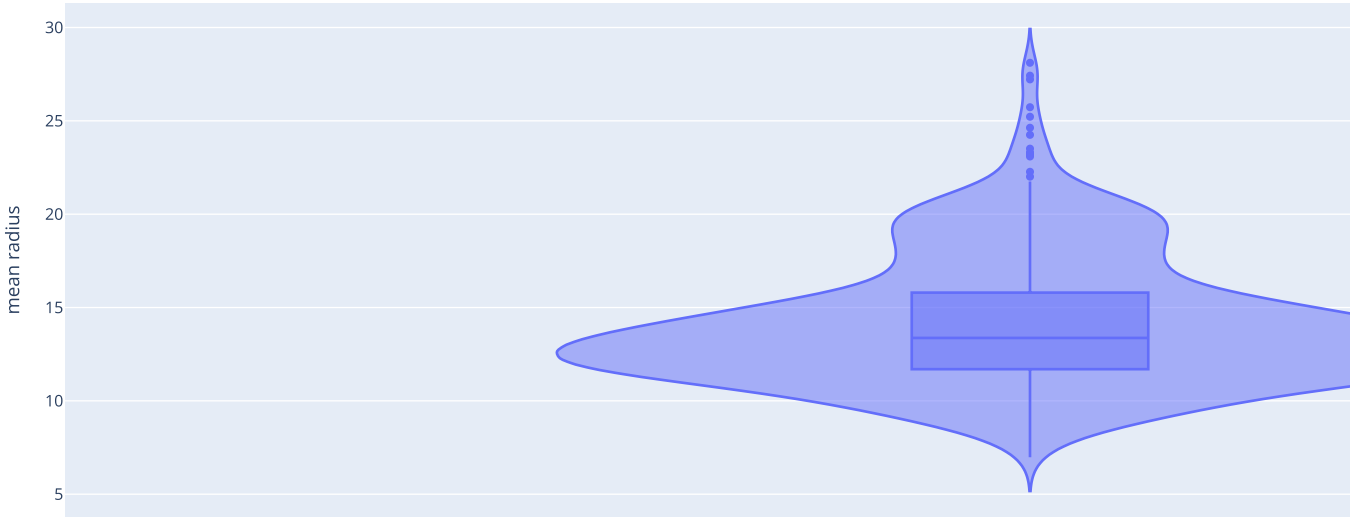

```
# A violin plot is a statistical representation of numerical data.
# It is similar to a box plot, with the addition of a rotated kernel density plot on each side.
fig=px.violin(cancer_df, y='mean radius')
fig.show()
```



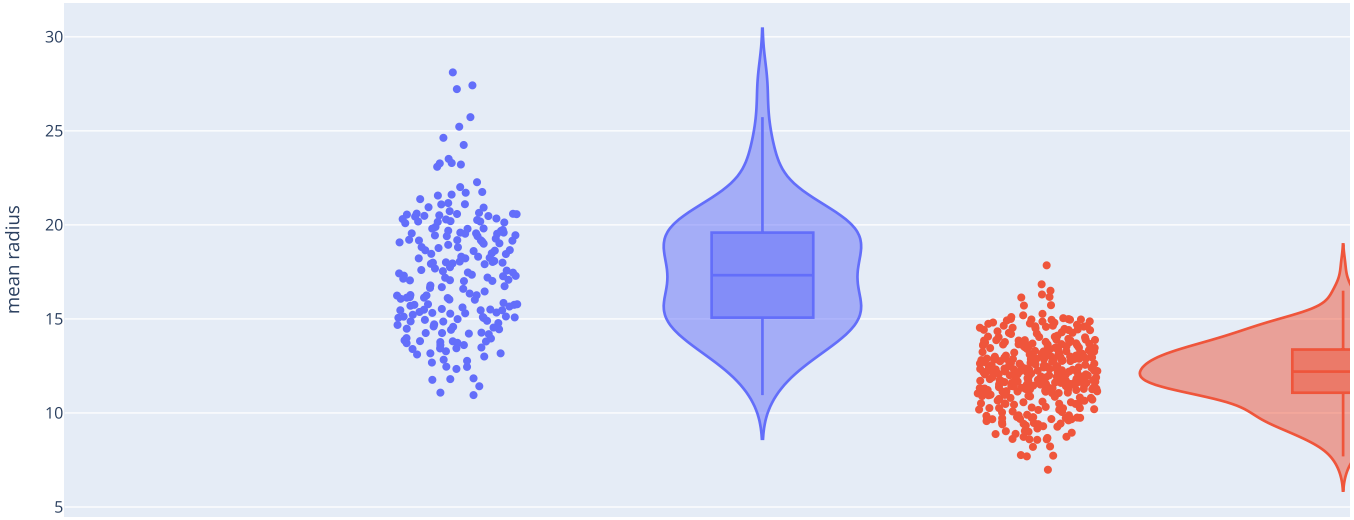

```
# Show data points
fig = px.violin(cancer_df, y = "mean radius", box = True, points = "all", hover_data = cancer_df.columns)
fig.show()
```



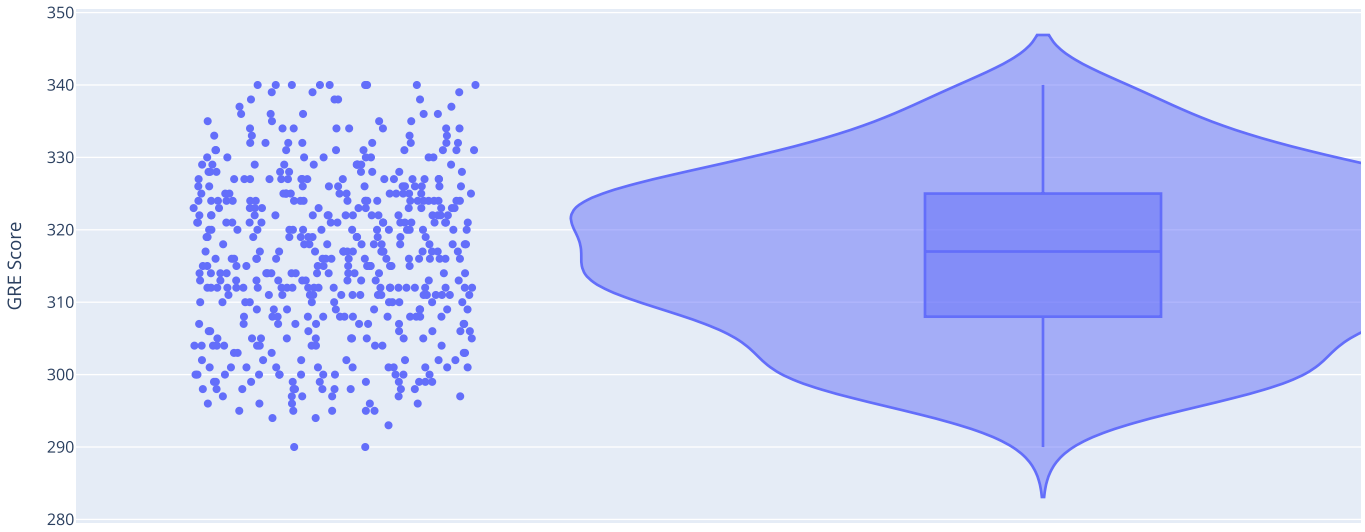

```
# Show data points and box plot
fig = px.violin(cancer_df, y = "mean radius", box = True, points = "outliers", hover_data = cancer_df.columns)
fig.show()
```




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




```
#Plot violin plot for GRE Score in university admission dataset
fig = px.violin(university_df, y = "GRE Score", box = True, points = "all", hover_data = university_df.columns)
fig.show()
```



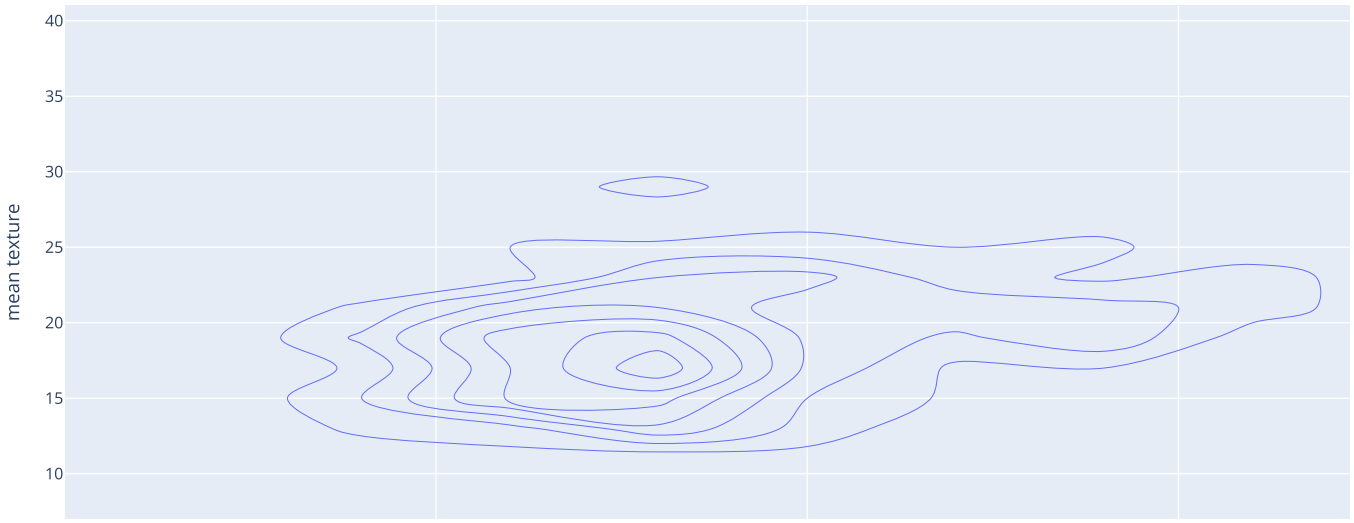

```
#Using the violin plot, what is the median value of the GRE Score?
university_df.median()
```

| | 0 |
|---|---|
| **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 |

```
#Calculate the mean value for GRE score and compare it to the 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 |

```
# A 2D histogram contour plot, also known as a density contour plot, is a 2-dimensional generalization of a histogram which rese
# and applying an aggregation function such as count or sum (if z is provided) to compute the value to be used to compute contou
# This kind of visualization (and the related 2D histogram, or density heatmap) is often used to manage over-plotting, or situat
fig = px.density_contour(cancer_df, x = "mean radius", y = "mean texture")
fig.show()
```




```
# 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()
```