Breeze Patel || Neu - Id 002821278 || Lab 8 - (Optional) - Data visualization using Python libraries

1. PLOT BASIC LINE PLOT

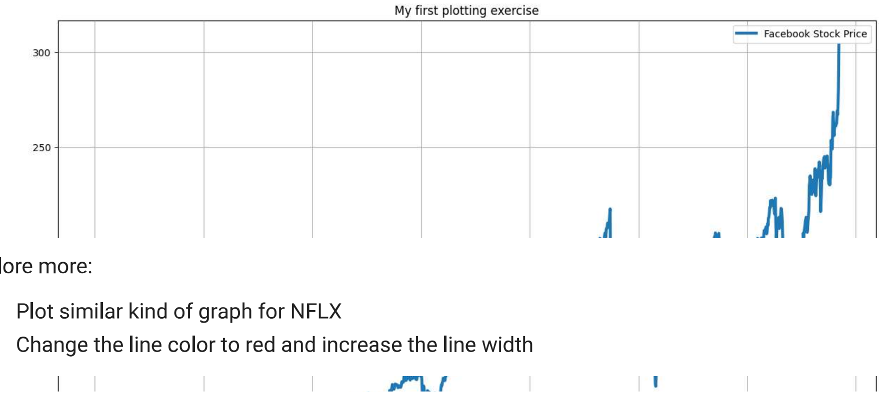
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```
# Read the stock prices data using pandas
stock_df = pd.read_csv('stock_data.csv')
stock_df
```

	Date	FB	TWTR	NFLX
0	2013-11-07	47.560001	44.900002	46.694286
1	2013-11-08	47.529999	41.650002	47.842857
2	2013-11-11	46.200001	42.900002	48.272858
3	2013-11-12	46.610001	41.900002	47.675713
4	2013-11-13	48.709999	42.599998	47.897144
...	...	...	...	...
1707	2020-08-20	269.010010	38.959999	497.899994
1708	2020-08-21	267.010010	39.259998	492.309998
1709	2020-08-24	271.390015	40.490002	488.809998
1710	2020-08-25	280.820007	40.549999	490.579987
1711	2020-08-26	303.910004	41.080002	547.530029

1712 rows × 4 columns

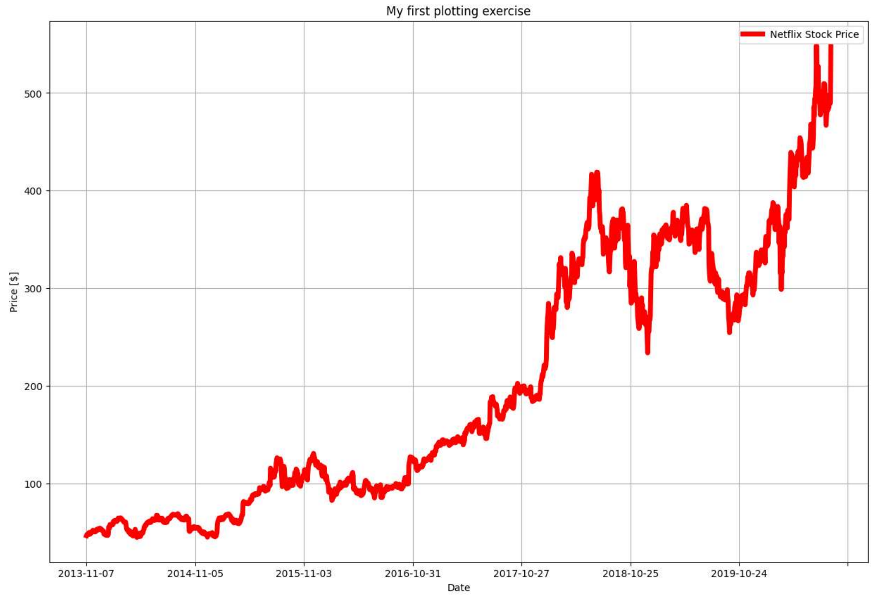
```
stock_df.plot(x = 'Date', y = 'FB', label = 'Facebook Stock Price', figsize = (15, 10), linewidth = 3)
plt.ylabel('Price [$]')
plt.title('My first plotting exercise')
plt.legend(loc = 'upper right')
plt.grid()
```



Explore more:

- Plot similar kind of graph for NFLX
- Change the line color to red and increase the line width

```
stock_df.plot(x = 'Date', y = 'NFLX', label = 'Netflix Stock Price', figsize = (15, 10), linewidth = 5, color = 'r')
plt.ylabel('Price [$]')
plt.title('My first plotting exercise')
plt.legend(loc = 'upper right')
plt.grid()
```



2. PLOT SCATTERPLOT

```
# Read daily return data using pandas
daily_return_df = pd.read_csv('stocks_daily_returns.csv')
daily_return_df
```

	Date	FB	TWTR	NFLX
0	2013-11-07	0.000000	0.000000	0.000000
1	2013-11-08	-0.063082	-7.238307	2.459768
2	2013-11-11	-2.798229	3.001200	0.898778
3	2013-11-12	0.887446	-2.331002	-1.237020
4	2013-11-13	4.505467	1.670635	0.464452
...	...	...	...	...
1707	2020-08-20	2.444881	0.179995	2.759374
1708	2020-08-21	-0.743467	0.770018	-1.122715
1709	2020-08-24	1.640390	3.132970	-0.710934

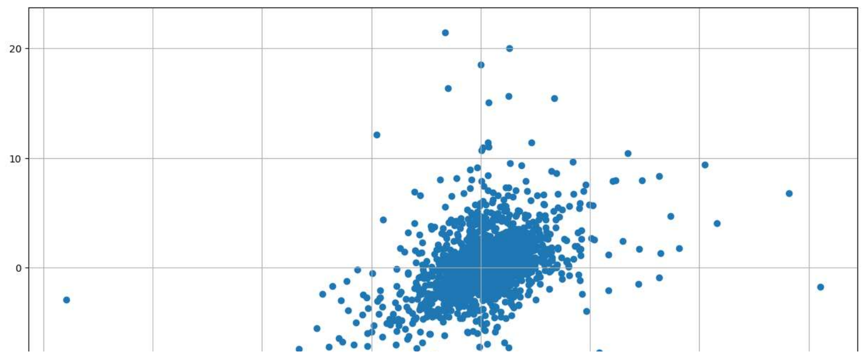
```
x = daily_return_df['FB']
x
```

```
0       0.000000
1      -0.063082
2      -2.798229
3       0.887446
4       4.505467
          ...
1707    2.444881
1708   -0.743467
1709    1.640390
1710    3.474701
1711    8.222348
Name: FB, Length: 1712, dtype: float64
```

```
y = daily_return_df['TWTR']
y
```

```
0       0.000000
1      -7.238307
2       3.001200
3      -2.331002
4       1.670635
          ...
1707    0.179995
1708    0.770018
1709    3.132970
1710    0.148177
1711    1.307036
Name: TWTR, Length: 1712, dtype: float64
```

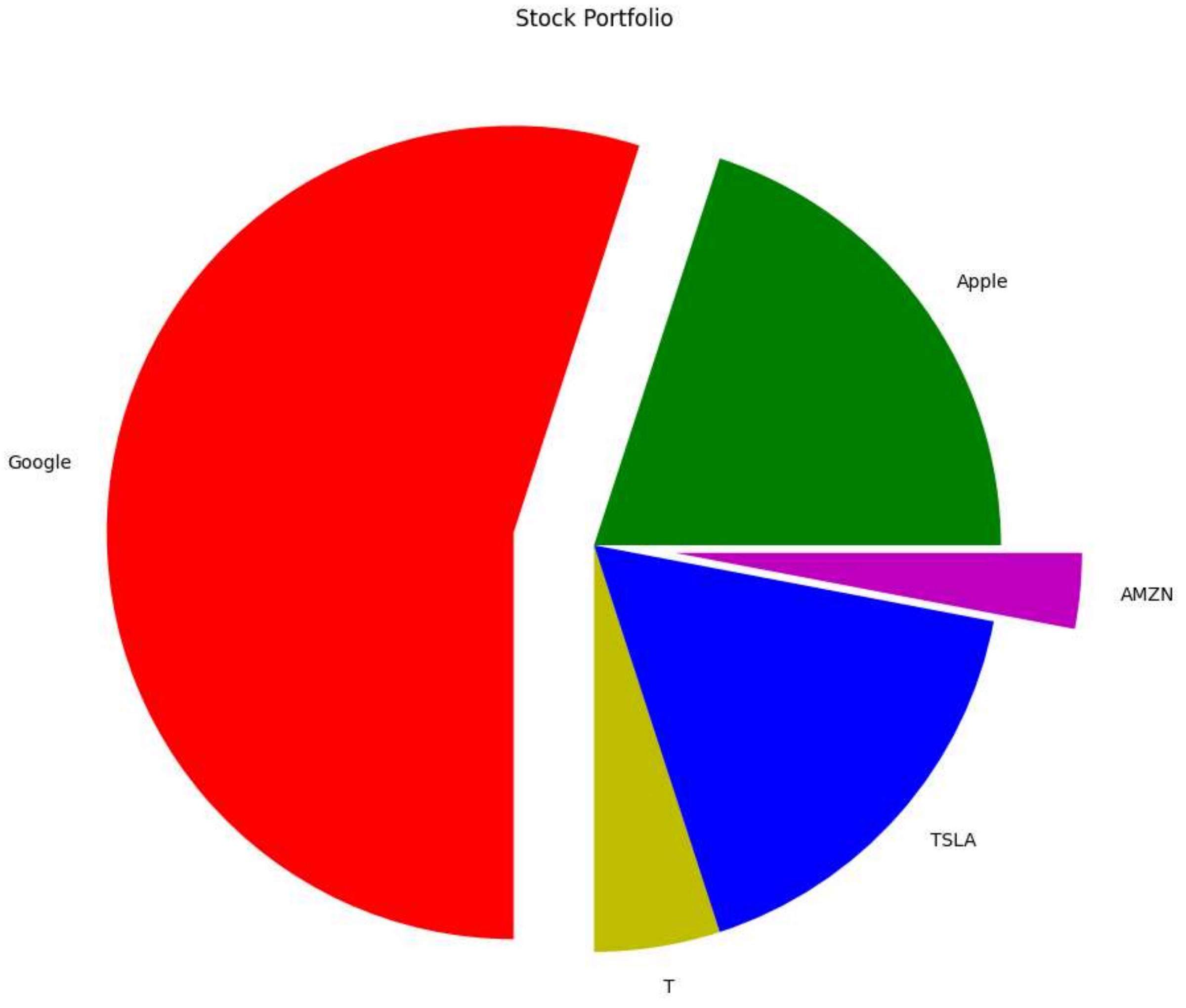
```
plt.figure(figsize = (15, 10))
plt.scatter(x, y)
plt.grid()
```



3. PLOT PIE CHART

```
values = [20, 55, 5, 17, 3]
colors = ['g', 'r', 'y', 'b', 'm']
labels = ["Apple", "Google", "T", "TSLA", "AMZN"]
explode = [0, 0.2, 0, 0, 0.2]
# Use matplotlib to plot a pie chart
plt.figure(figsize = (10, 10))
plt.pie(values, colors = colors, labels = labels, explode = explode)
plt.title('Stock Portfolio')
```

```
Text(0.5, 1.0, 'Stock Portfolio')
```

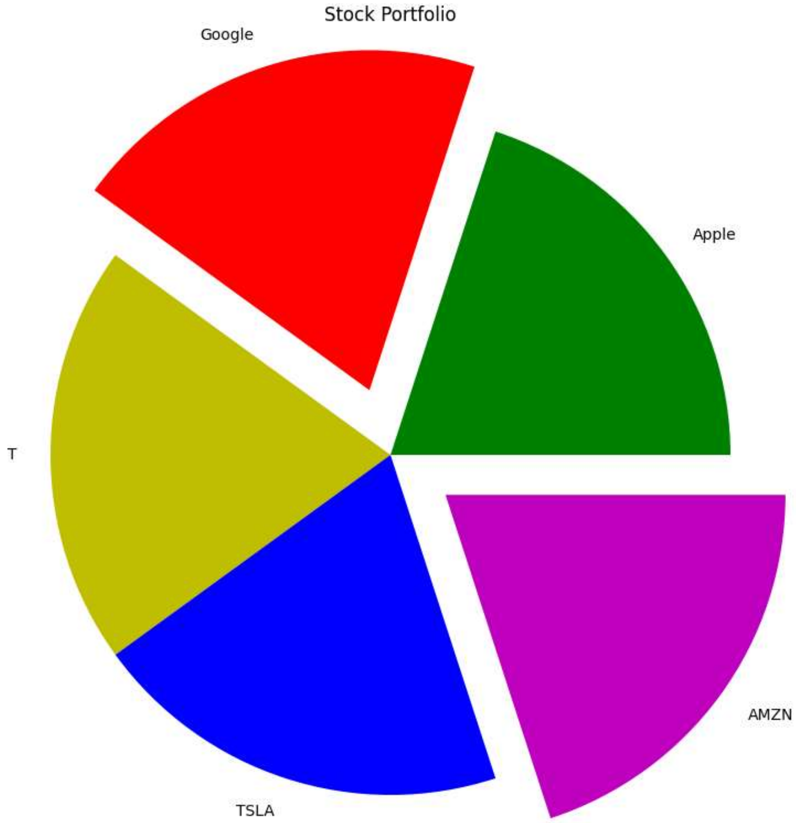


Explore more:

- Plot the pie chart for the same stocks assuming equal allocation
- Explode Amazon and Google slices

```
values = [20, 20, 20, 20, 20]
colors = ['g', 'r', 'y', 'b', 'm']
labels = ["Apple", "Google", "T", "TSLA", "AMZN"]
explode = [0, 0.2, 0, 0, 0.2]
# Use matplotlib to plot a pie chart
plt.figure(figsize = (10, 10))
plt.pie(values, colors = colors, labels = labels, explode = explode)
plt.title('Stock Portfolio')
```

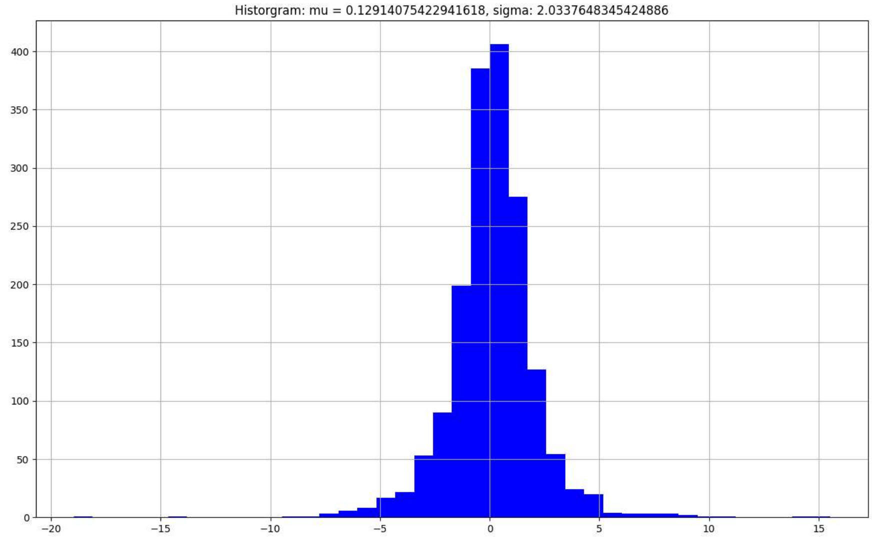
```
Text(0.5, 1.0, 'Stock Portfolio')
```



4. PLOT HISTOGRAMS

```
# A histogram represents data using bars of various heights.
# Each bar groups numbers inato specific ranges.
# Taller bars show that more data falls within that specific range.
mu = daily_return_df['FB'].mean()
sigma = daily_return_df['FB'].std()

num_bins = 40
plt.figure(figsize = (15, 9))
plt.hist(daily_return_df['FB'], num_bins, facecolor = 'blue'); # ; is to get rid of extra text printing
plt.grid()

plt.title('Historgram: mu = ' + str(mu) + ', sigma: ' + str(sigma))
```

```
Text(0.5, 1.0, 'Historgram: mu = 0.12914075422941618, sigma: 2.0337648345424886')
```



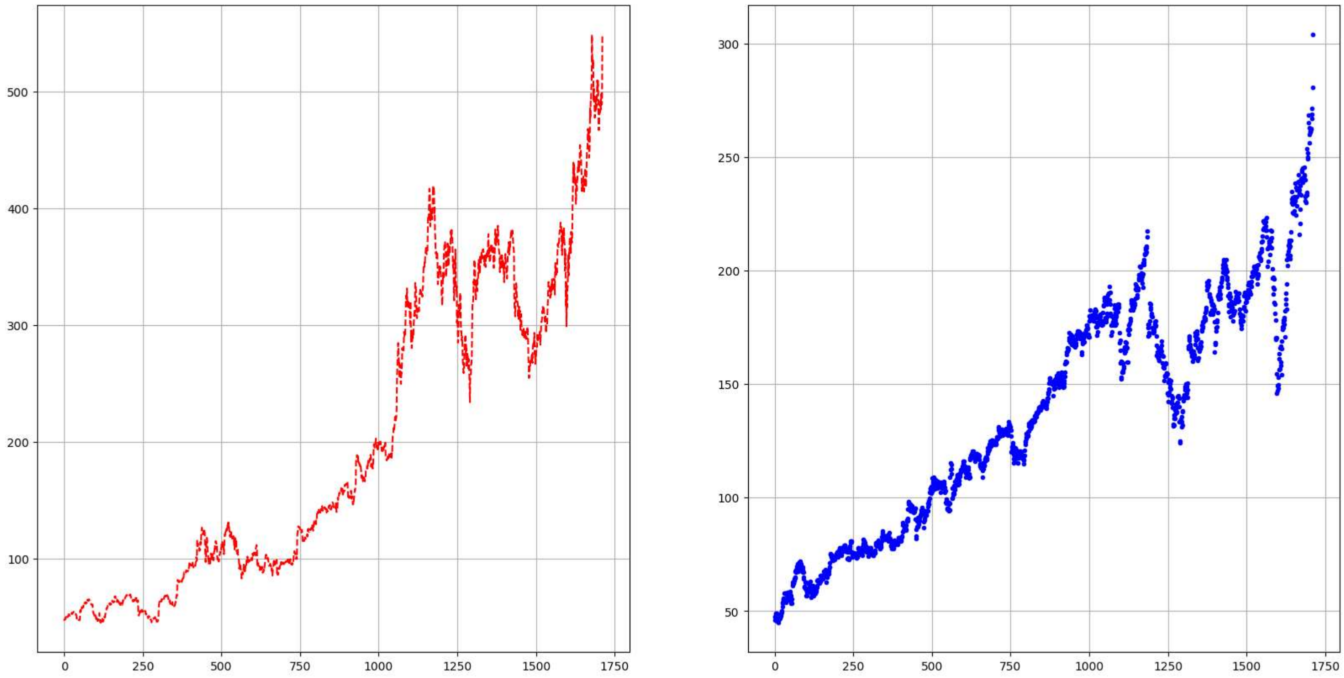
5. PLOT MULTIPLE PLOTS

```
stock_df.plot(x = 'Date', y = ['NFLX', 'FB', 'TWTR'], figsize = (18, 10), linewidth = 3)
plt.ylabel('price [$]')
plt.title('Stock Prices')
plt.grid()
plt.legend(loc = 'upper center')
```

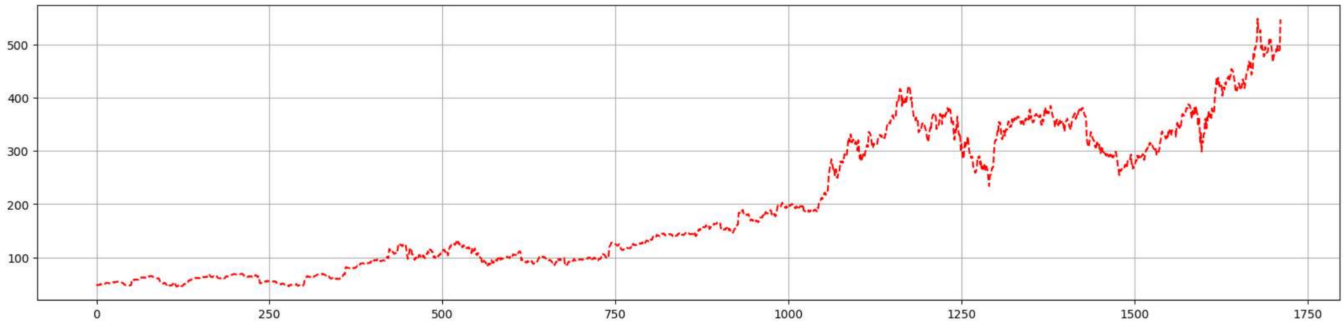
```
<matplotlib.legend.Legend at 0x7fb6cc435fc0>
```

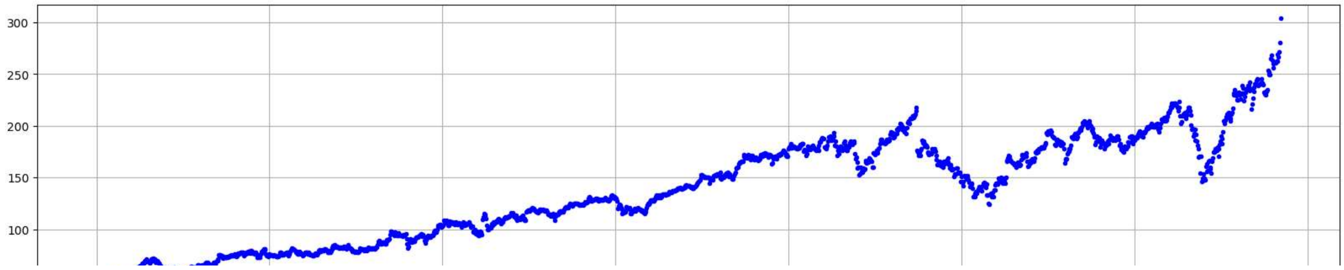
6. PLOT SUBPLOTS

```
plt.figure(figsize = (20, 10))

plt.subplot(1, 2, 1) # will have 1 row and 2 columns, we are plotting first one
plt.plot(stock_df['NFLX'], 'r--') # r color, -- style
plt.grid()

plt.subplot(1, 2, 2) # will have 1 row and 2 columns, we are plotting second one
plt.plot(stock_df['FB'], 'b.')
plt.grid()
```




```
plt.figure(figsize = (20, 10))

plt.subplot(2, 1, 1) # will have 2 rows and 1 column, we are plotting first one
plt.plot(stock_df['NFLX'], 'r--') # r color, -- style
plt.grid()

plt.subplot(2, 1, 2) # will have 2 rows and 1 column, we are plotting second one
plt.plot(stock_df['FB'], 'b.')
plt.grid()
```

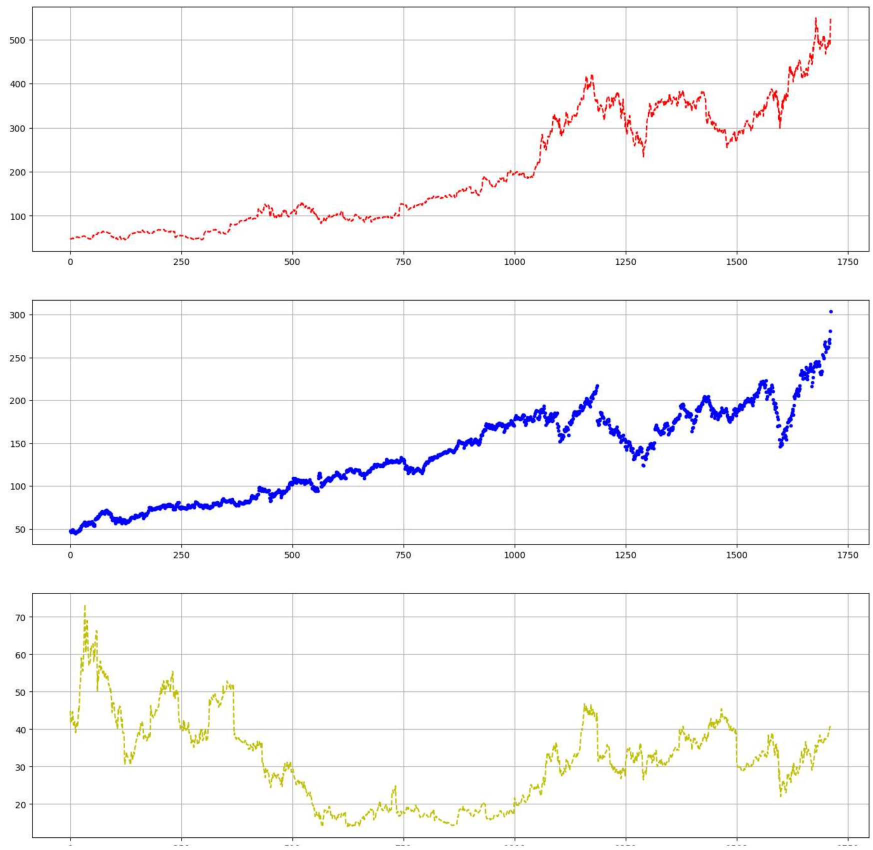




Explore more:

- Create subplots like above for Twitter, Facebook and Netflix

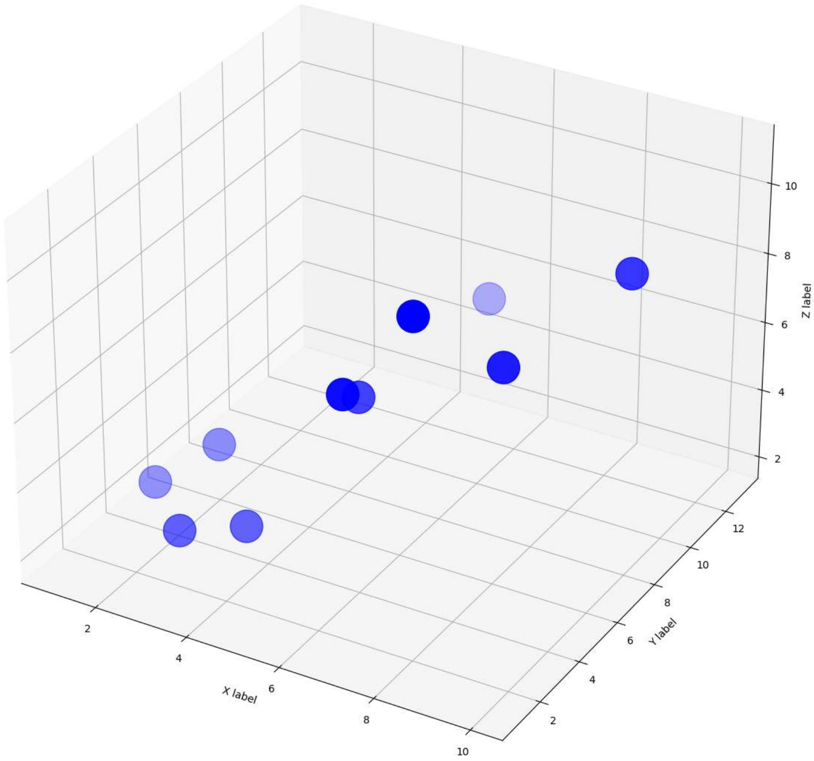
```
plt.figure(figsize = (17, 17))

plt.subplot(3, 1, 1) # will have 2 rows and 1 column, we are plotting first one
plt.plot(stock_df['NFLX'], 'r--') # r color, -- style
plt.grid()

plt.subplot(3, 1, 2) # will have 2 rows and 1 column, we are plotting second one
plt.plot(stock_df['FB'], 'b.')
plt.grid()

plt.subplot(3, 1, 3) # will have 2 rows and 1 column, we are plotting second one
plt.plot(stock_df['TWTR'], 'y--')
plt.grid()
```



7. PLOT 3D PLOTS

```
# Toolkits are collections of application-specific functions that extend Matplotlib.
# mpl_toolkits.mplot3d provides tools for basic 3D plotting.
# https://matplotlib.org/mpl_toolkits/index.html

from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure(figsize = (15, 15))
ax = fig.add_subplot(111, projection = '3d')

x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
y = [5, 6, 2, 3, 13, 4, 1, 2, 4, 8]
z = [2, 3, 3, 3, 5, 7, 9, 11, 9, 10]

ax.scatter(x, y, z, c = 'b', s = 1000) # c for color, s for size of each points
ax.set_xlabel('X label')
ax.set_ylabel('Y label')
ax.set_zlabel('Z label')
```

```
Text(0.5, 0, 'Z label')
```

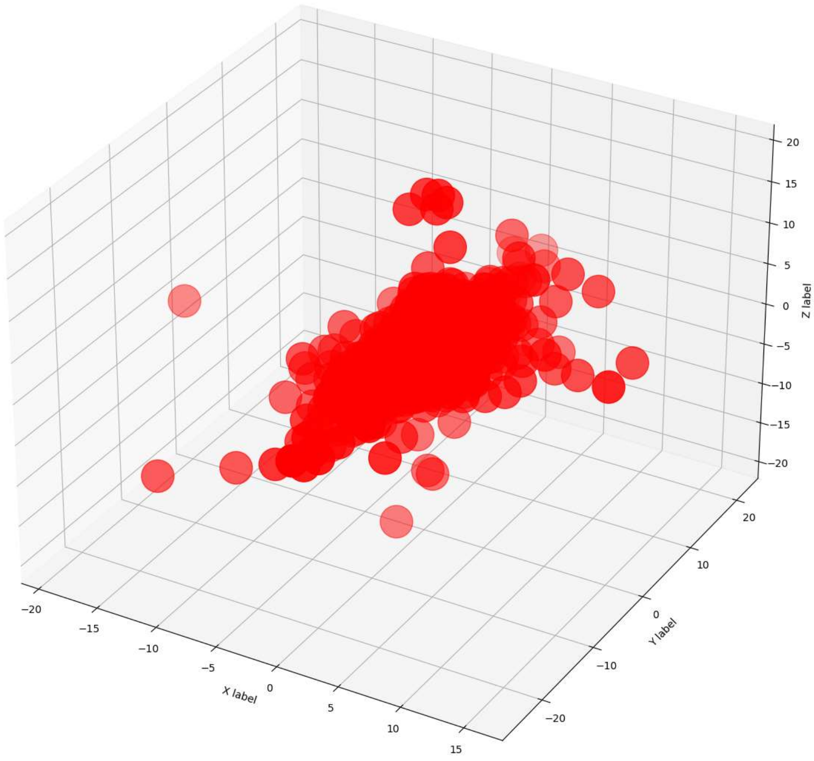


Explore more:

- Create a 3D plot with daily return values of Twitter, Facebook and Netflix

```
fig = plt.figure(figsize = (15, 15))
ax = fig.add_subplot(111, projection = '3d')

x = daily_return_df['FB'].tolist()
y = daily_return_df['TWTR'].tolist()
z = daily_return_df['NFLX'].tolist()

ax.scatter(x, y, z, c = 'r', s = 1000) # c for color, s for size of each points
ax.set_xlabel('X label')
ax.set_ylabel('Y label')
ax.set_zlabel('Z label')
```

```
Text(0.5, 0, 'Z label')
```



8. SEABRON SCATTERPLOT & COUNTPLOT

```
# Seaborn is a visualization library that sits on top of matplotlib
# Seaborn offers enhanced features compared to matplotlib
# https://seaborn.pydata.org/examples/index.html

# import libraries
import seaborn as sns # Statistical data visualization
```

```
# Import Cancer data drom the Sklearn library
from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()
cancer
```

```
0.802  542.2\n    smoothness (standard error):          0.002  0.031\n    compactness (standard error):         0.002  0.135\n
concavity (standard error):           0.0    0.396\n    concave points (standard error):      0.0    0.053\n    symmetry
(standard error):            0.008  0.079\n    fractal dimension (standard error):   0.001  0.03\n    radius (worst):
7.93   36.04\n    texture (worst):                      12.02  49.54\n    perimeter (worst):                    50.41  251.2\n
area (worst):                         185.2  4254.0\n    smoothness (worst):                   0.071  0.223\n    compactness
(worst):                 0.027  1.058\n    concavity (worst):                    0.0    1.252\n    concave points (worst):
0.0    0.291\n    symmetry (worst):                     0.156  0.664\n    fractal dimension (worst):            0.055  0.208\n
===================================== ====== ======\n\n    :Missing Attribute Values: None\n\n    :Class Distribution: 212 -
Malignant, 357 - Benign\n\n    :Creator:  Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian\n\n    :Donor: Nick
Street\n\n    :Date: November, 1995\n\nThis is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic)
datasets.\nhttps://goo.gl/U2Uwz2\n\nFeatures are computed from a digitized image of a fine needle\naspirate (FNA) of a breast
mass.  They describe\ncharacteristics of the cell nuclei present in the image.\n\nSeparating plane described above was obtained
using\nMultisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree\nConstruction Via Linear Programming." Proceedings of the
4th\nMidwest Artificial Intelligence and Cognitive Science Society,\npp. 97-101, 1992], a classification method which uses
linear\nprogramming to construct a decision tree.  Relevant features\nwere selected using an exhaustive search in the space of
1-4\nfeatures and 1-3 separating planes.\n\nThe actual linear program used to obtain the separating plane\nin the 3-dimensional
space is that described in:\n[K. P. Bennett and O. L. Mangasarian: "Robust Linear\nProgramming Discrimination of Two Linearly
Inseparable Sets",\nOptimization Methods and Software 1, 1992, 23-34].\n\nThis database is also available through the UW CS ftp
server:\n\nftp ftp.cs.wisc.edu\ncd math-prog/cpo-dataset/machine-learn/WDBC/\n\n.. topic:: References\n\n   - W.N. Street, W.H.
Wolberg and O.L. Mangasarian. Nuclear feature extraction \n     for breast tumor diagnosis. IS&T/SPIE 1993 International
Symposium on \n     Electronic Imaging: Science and Technology, volume 1905, pages 861-870,\n     San Jose, CA, 1993.\n   - O.L.
Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and \n     prognosis via linear programming. Operations
Research, 43(4), pages 570-577, \n     July-August 1995.\n   - W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning
techniques\n     to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77 (1994) \n     163-171.',
 'feature_names': array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
        'mean smoothness', 'mean compactness', 'mean concavity',
        'mean concave points', 'mean symmetry', 'mean fractal dimension',
        'radius error', 'texture error', 'perimeter error', 'area error',
        'smoothness error', 'compactness error', 'concavity error',
        'concave points error', 'symmetry error',
```

```
# Create a dataFrame named df_cancer with input/output data
df_cancer = pd.DataFrame(np.c_[cancer['data'], cancer['target']], columns = np.append(cancer['feature_names'], ['target']))
```

```
# Check out the head of the dataframe
df_cancer
```

	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	0.30010	0.14710	0.2419	0.07871	...	17.33	184.60
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.05667	...	23.41	158.80
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.05999	...	25.53	152.50
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.09744	...	26.50	98.87
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.05883	...	16.67	152.20
...	...	...	...	...	...	...	...	...	...	...	...	...	...
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623	...	26.40	166.10
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533	...	38.25	155.00
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648	...	34.12	126.70
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016	...	39.42	184.60
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884	...	30.37	59.16

569 rows × 31 columns

```
# Check out the heaf of the dataframe
df_cancer.head(7)
```

	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	wor ar
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	201
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	195
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	170
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	56
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	157
5	12.45	15.70	82.57	477.1	0.12780	0.17000	0.1578	0.08089	0.2087	0.07613	...	23.75	103.40	74
6	18.25	19.98	119.60	1040.0	0.09463	0.10900	0.1127	0.07400	0.1794	0.05742	...	27.66	153.20	160

7 rows × 31 columns

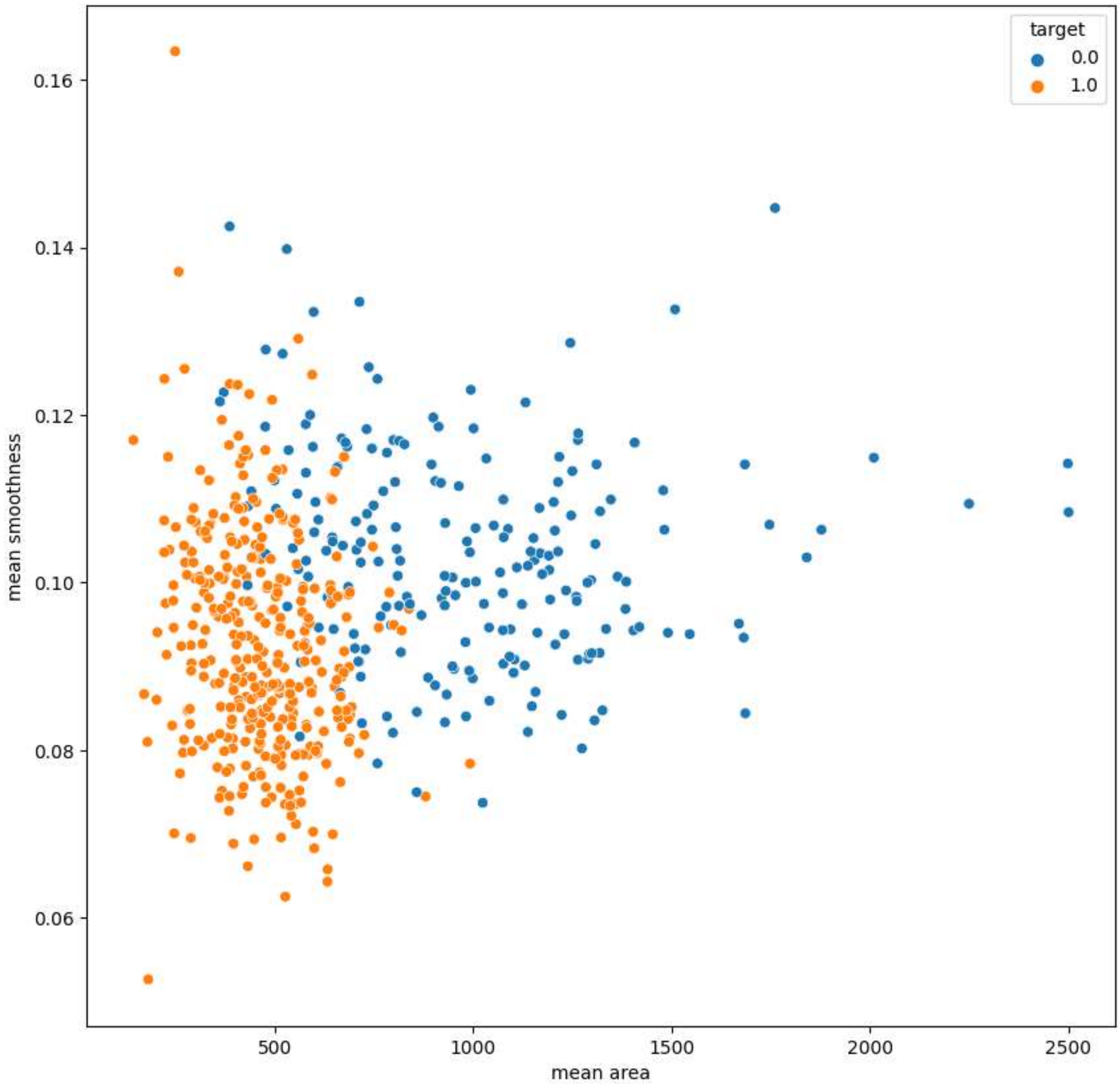
```
# Check out the tail of the dataframe
df_cancer.tail(7)
```

	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	w
562	15.22	30.62	103.40	716.9	0.10480	0.20870	0.25500	0.09429	0.2128	0.07152	...	42.79	128.70	9
563	20.92	25.09	143.00	1347.0	0.10990	0.22360	0.31740	0.14740	0.2149	0.06879	...	29.41	179.10	18
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623	...	26.40	166.10	20
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533	...	38.25	155.00	17
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648	...	34.12	126.70	1
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016	...	39.42	184.60	18
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884	...	30.37	59.16	2

7 rows × 31 columns

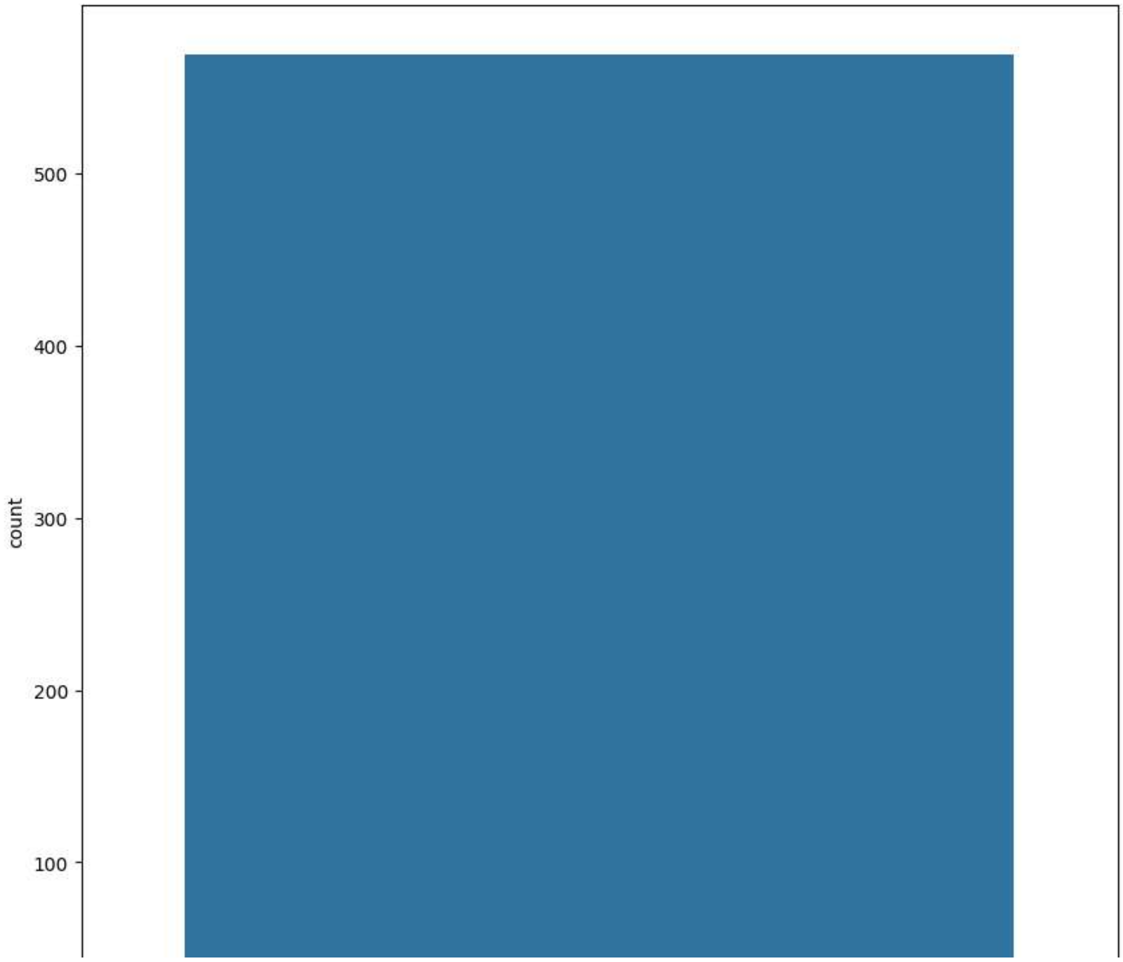
```
# Plot scatter plot between mean area and mean smoothness
plt.figure(figsize = (10,10))
sns.scatterplot(x = 'mean area', y = 'mean smoothness', hue = 'target', data = df_cancer)
```

```
<Axes: xlabel='mean area', ylabel='mean smoothness'>
```




```
# Let's print out countplot to know how many samples belong to class #0 and #1
plt.figure(figsize = (10,10))
sns.countplot(df_cancer['target'], label = 'Count')
```

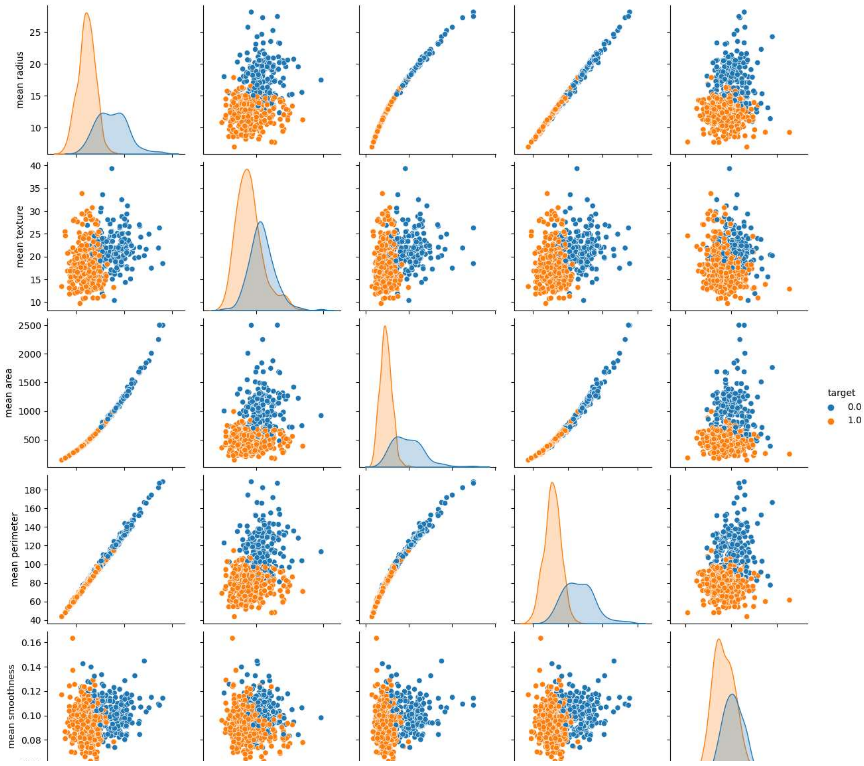
<Axes: ylabel='count'>



9. SEABORN PAIRPLOT, DISPLOT, AND HEATMAPS/CORRELATIONS

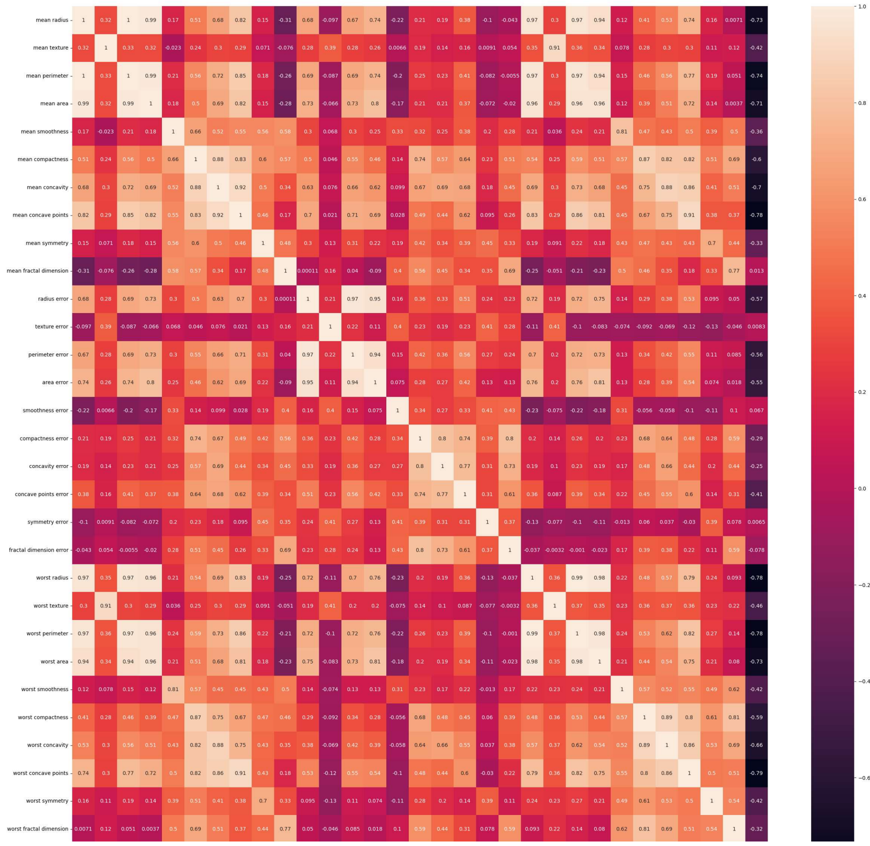
```
# Plot the pairplot
sns.pairplot(df_cancer, hue = 'target', vars = ['mean radius', 'mean texture', 'mean area', 'mean perimeter', 'mean smoothness'])
```

<seaborn.axisgrid.PairGrid at 0x7fb6fe3c63e0>



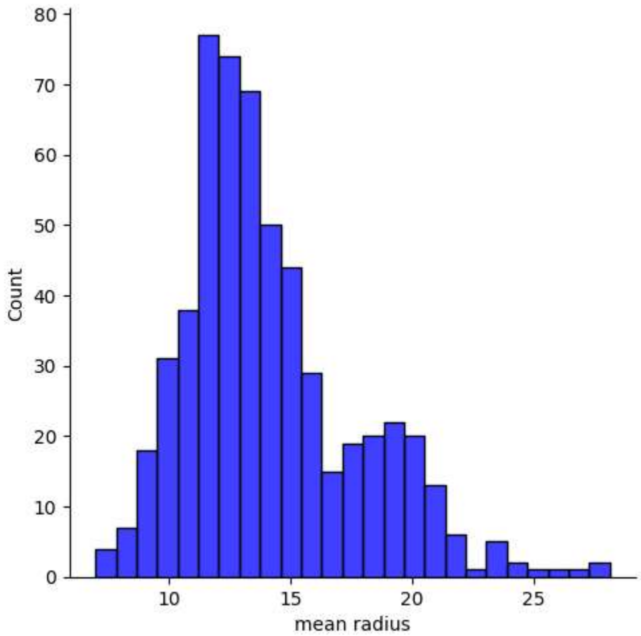

```
# Strong correlation between the mean radius and mean perimeter, mean area and mean primeter
plt.figure(figsize = (30, 30))
sns.heatmap(df_cancer.corr(), annot = True)
```

```
<Axes: >
```



```
# plot the distplot
# Displot combines matplotlib histogram function with kdeplot() (Kernel density estimate)
# KDE is used to plot the Probability Density of a continuous variable.

sns.displot(df_cancer['mean radius'], bins = 25, color = 'b')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fb6be6197e0>
```



Explore more:

- Plot two separate distplot for each target class #0 and target class #1

```
class_0_df = df_cancer[df_cancer['target'] == 0]
class_1_df = df_cancer[df_cancer['target'] == 1]
```

```
class_0_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	worst perimeter	w
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	0.07871	...	17.33	184.60	2
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.05667	...	23.41	158.80	1
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.05999	...	25.53	152.50	1
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.09744	...	26.50	98.87	[illegible]
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.05883	...	16.67	152.20	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
563	20.92	25.09	143.00	1347.0	0.10990	0.22360	0.31740	0.14740	0.2149	0.06879	...	29.41	179.10	1
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623	...	26.40	166.10	2
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533	...	38.25	155.00	1
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648	...	34.12	126.70	1
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016	...	39.42	184.60	1

```
class_1_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	worst perimeter	wo a
19	13.540	14.36	87.46	566.3	0.09779	0.08129	0.06664	0.047810	0.1885	0.05766	...	19.26	99.70	7
20	13.080	15.71	85.63	520.0	0.10750	0.12700	0.04568	0.031100	0.1967	0.06811	...	20.49	96.09	6
21	9.504	12.44	60.34	273.9	0.10240	0.06492	0.02956	0.020760	0.1815	0.06905	...	15.66	65.13	3
37	13.030	18.42	82.61	523.8	0.08983	0.03766	0.02562	0.029230	0.1467	0.05863	...	22.81	84.46	5
46	8.196	16.84	51.71	201.9	0.08600	0.05943	0.01588	0.005917	0.1769	0.06503	...	21.96	57.26	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
558	14.590	22.68	96.39	657.1	0.08473	0.13300	0.10290	0.037360	0.1454	0.06147	...	27.27	105.90	7
559	11.510	23.93	74.52	403.5	0.09261	0.10210	0.11120	0.041050	0.1388	0.06570	...	37.16	82.28	4
560	14.050	27.15	91.38	600.4	0.09929	0.11260	0.04462	0.043040	0.1537	0.06171	...	33.17	100.20	7
561	11.200	29.37	70.67	386.0	0.07449	0.03558	0.00000	0.000000	0.1060	0.05502	...	38.30	75.19	4
568	7.760	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.000000	0.1587	0.05884	...	30.37	59.16	2

357 rows × 31 columns

```
plt.figure(figsize = (10, 7))
sns.displot(class_0_df['mean radius'], bins = 25, color = 'blue')
sns.displot(class_1_df['mean radius'], bins = 25, color = 'red')
plt.grid()
```