# 1. Plot basic line plot

In [ ]:

```
from jupyterthemes import jtplot
jtplot.style(theme = 'monokai', context = 'notebook', ticks = True, grid = False)
# setting the style of the notebook to be monokai theme
# this line of code is important to ensure that we are able to see the x and y axes cle
# If you don't run this code line, you will notice that the xlabel and ylabel on any pl
```

In [1]:

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

In [13]:

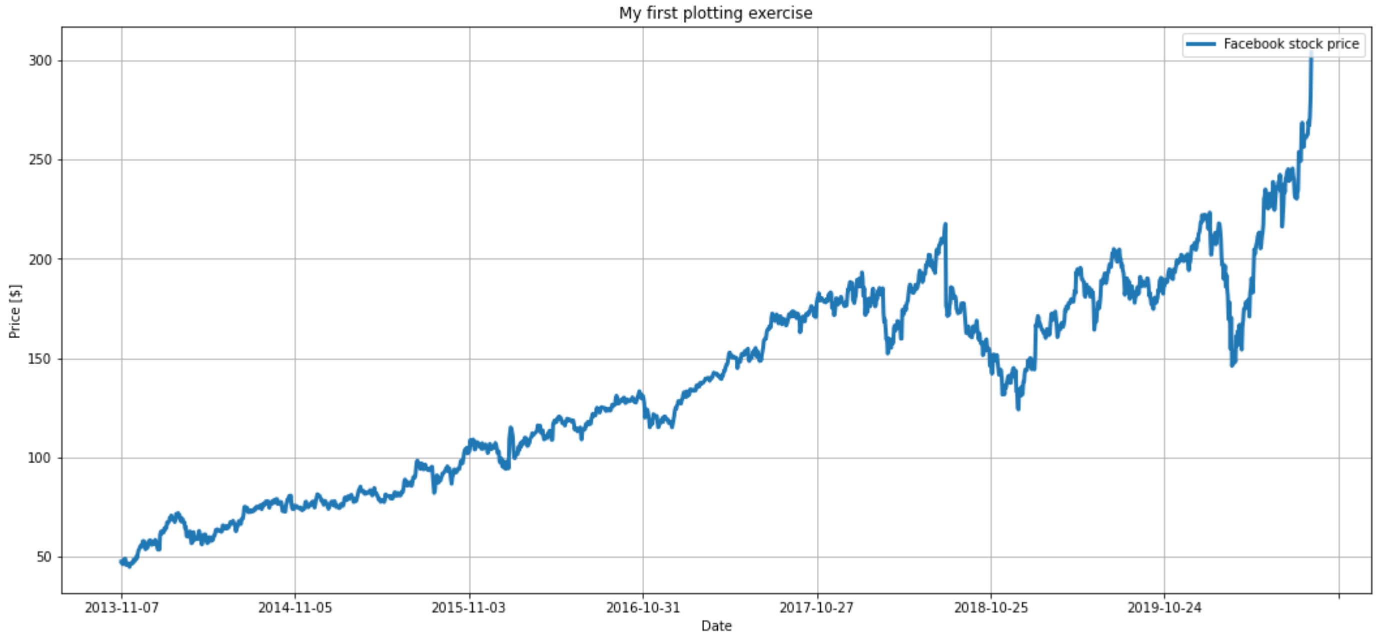
```
# read the stock prices data using pandas
stock_df = pd.read_csv('C:/Users/lenovo/Desktop/pyhton project/stock_data.csv')
df.head(10)
```

Out[13]:

| | 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 |
| **5** | 2013-11-14 | 48.990002 | 44.689999 | 48.938572 |
| **6** | 2013-11-15 | 49.009998 | 43.980000 | 49.965714 |
| **7** | 2013-11-18 | 45.830002 | 41.139999 | 48.824287 |
| **8** | 2013-11-19 | 46.360001 | 41.750000 | 48.184284 |
| **9** | 2013-11-20 | 46.430000 | 41.049999 | 48.502857 |

In [14]:

```
stock_df.plot(x = 'Date', y = 'FB', label = 'Facebook stock price', figsize= (18,8), li
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

In [15]:

```
stock_df.plot(x = 'Date', y = 'NFLX', label = 'Netflix Stock Price', figsize = (18, 8),
plt.ylabel('Price [$]')
plt.title('My first plotting exercise')
plt.legend(loc = 'upper right')
plt.grid()
```

Loading [MathJax]/extensions/Safe.js


# 2. PLOT SCATTERPLOT

In [16]:

```
# Read daily return data using pandas
daily_return_df = pd.read_csv('C:/Users/lenovo/Desktop/pyhton project/stocks_daily_retu
daily_return_df.head(10)
```

Out[16]:

| | 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 |
| **5** | 2013-11-14 | 0.574837 | 4.906106 | 2.174301 |
| **6** | 2013-11-15 | 0.040816 | -1.588720 | 2.098839 |
| **7** | 2013-11-18 | -6.488464 | -6.457483 | -2.284420 |
| **8** | 2013-11-19 | 1.156446 | 1.482744 | -1.310829 |
| **9** | 2013-11-20 | 0.150990 | -1.676649 | 0.661155 |

In [19]:

```
x = daily_return_df['FB']
x.head(10)
```

Out[19]:

```
0    0.000000
1   -0.063082
2   -2.798229
3    0.887446
4    4.505467
5    0.574837
6    0.040816
7   -6.488464
8    1.156446
9    0.150990
Name: FB, dtype: float64
```

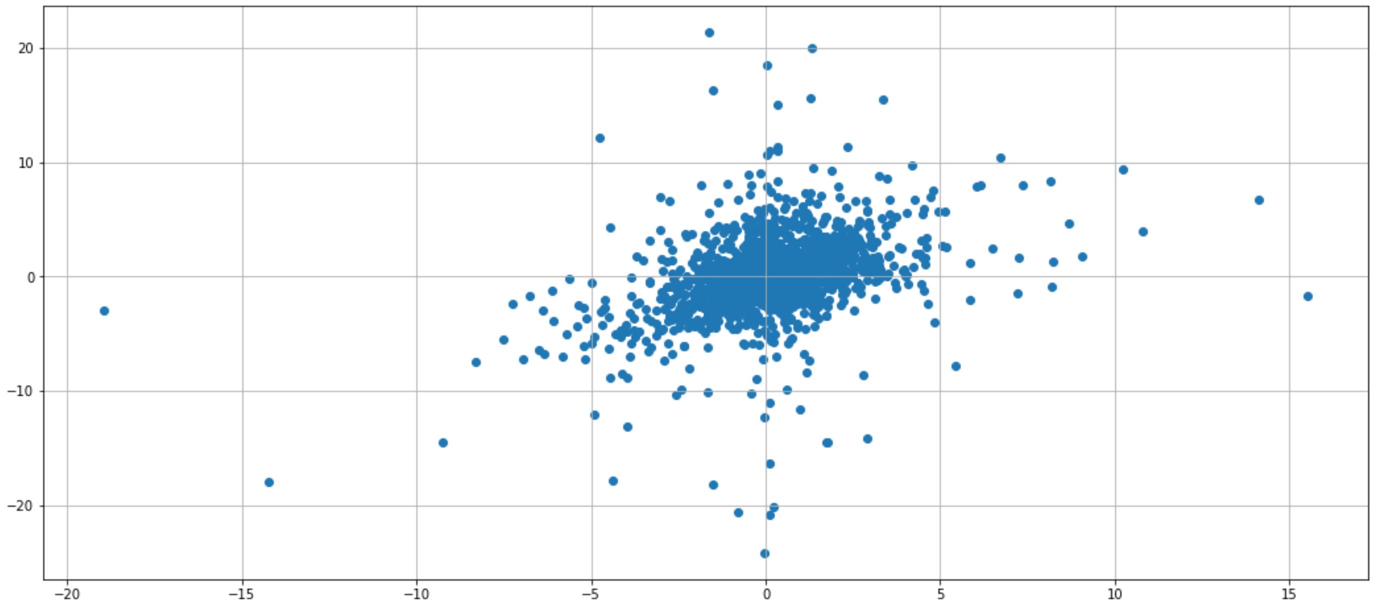
In [20]:

```
y = daily_return_df['TWTR']
y.head(10)
```

Out[20]:

```
0    0.000000
1   -7.238307
2    3.001200
3   -2.331002
4    1.670635
5    4.906106
6   -1.588720
7   -6.457483
8    1.482744
9   -1.676649
Name: TWTR, dtype: float64
```

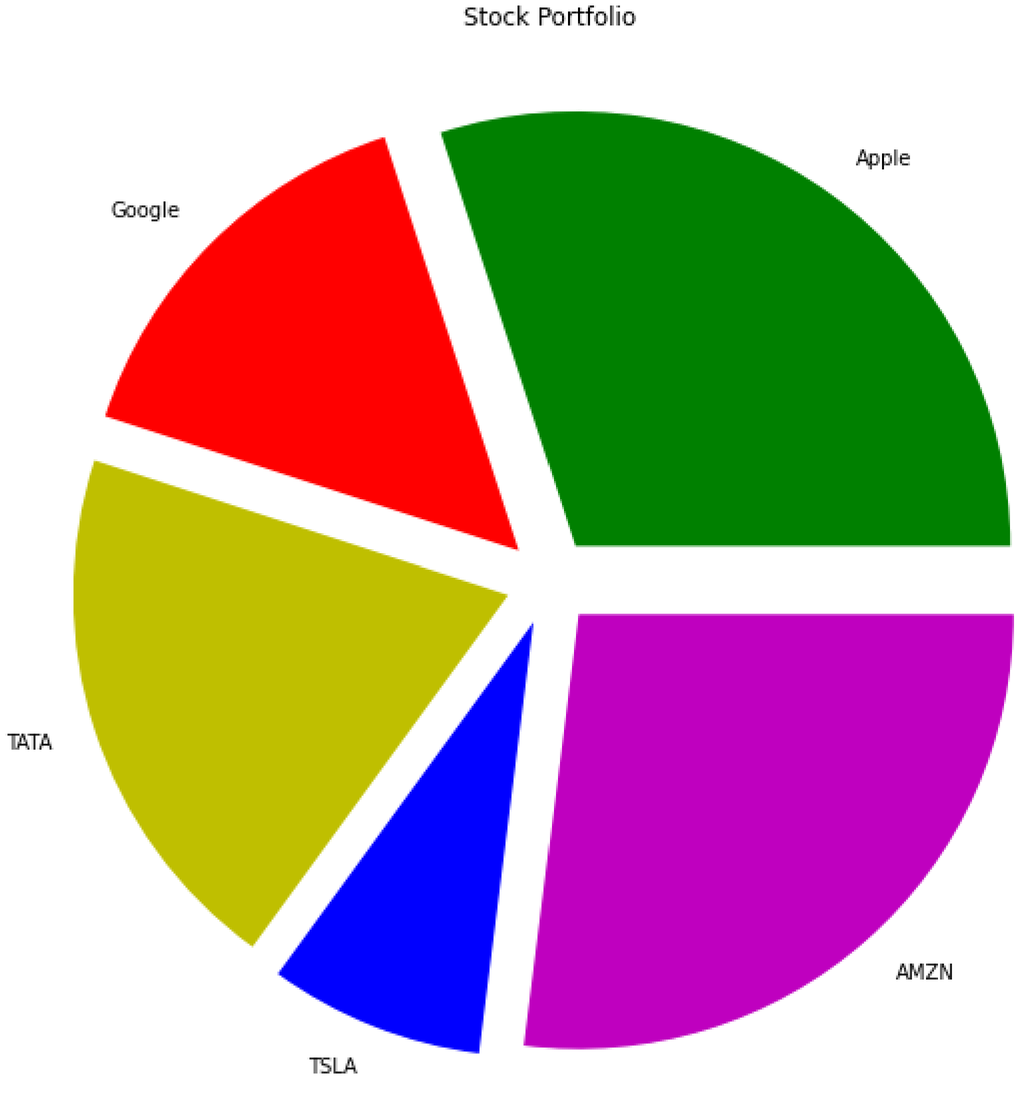
In [21]:
```
plt.figure(figsize = (18,8))
plt.scatter(x,y)
plt.grid()
```




# 3. PLOT PIE CHART

In [27]:
```
values = [30, 15, 20, 8, 27]  # total 100
colors = ['g', 'r', 'y', 'b', 'm']
labels = ["Apple", "Google", "TATA", "TSLA", "AMZN"]
explode = [0.1, 0.1, 0.1, 0.1, 0.1]
# 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')
```

Out[27]:
```
Text(0.5, 1.0, 'Stock Portfolio')
```




Explore more:

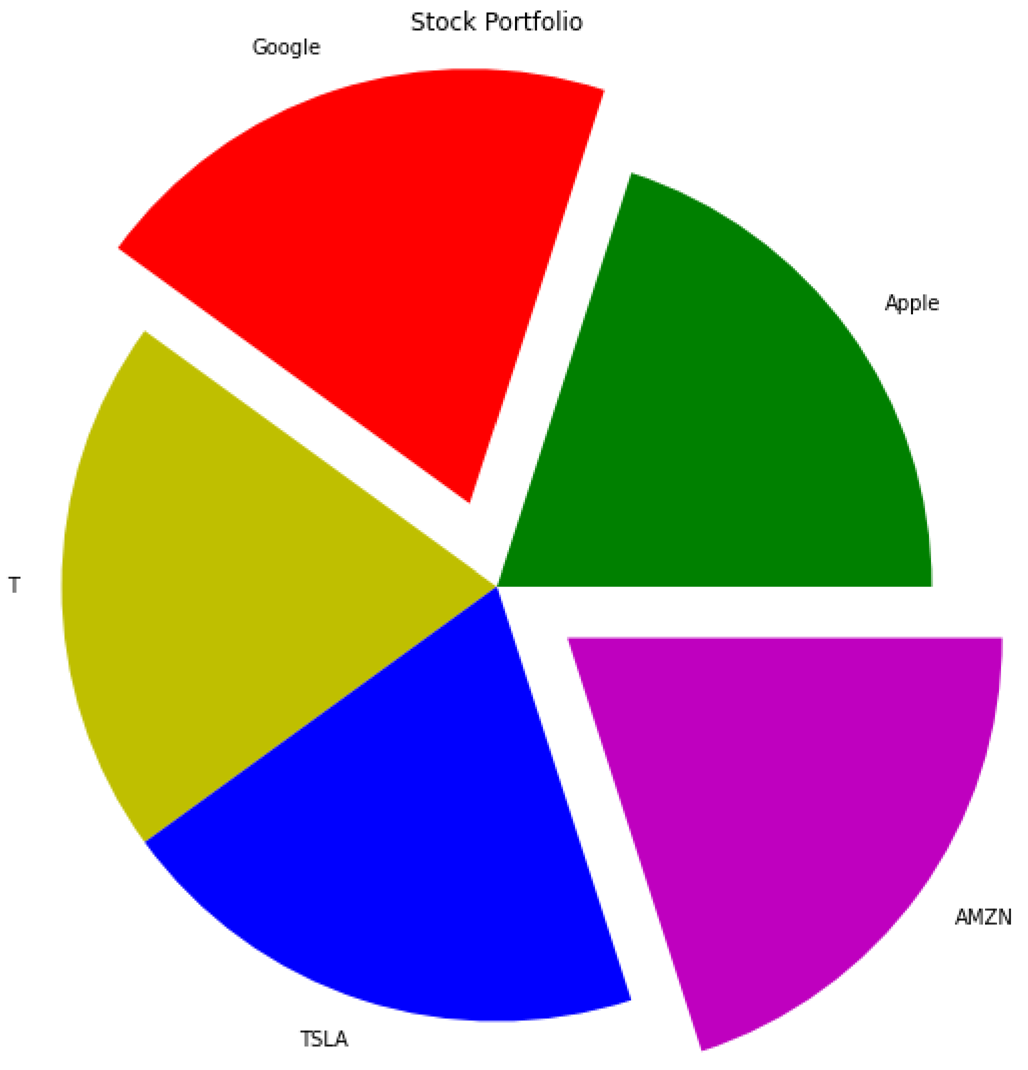
.Plot the pie chart for the same stocks assuming equal allocation

Loading [MathJax]/extensions/Safe.js

.Explode Amazon and Google slices

In [28]:
```
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')
```

Out[28]: Text(0.5, 1.0, 'Stock Portfolio')

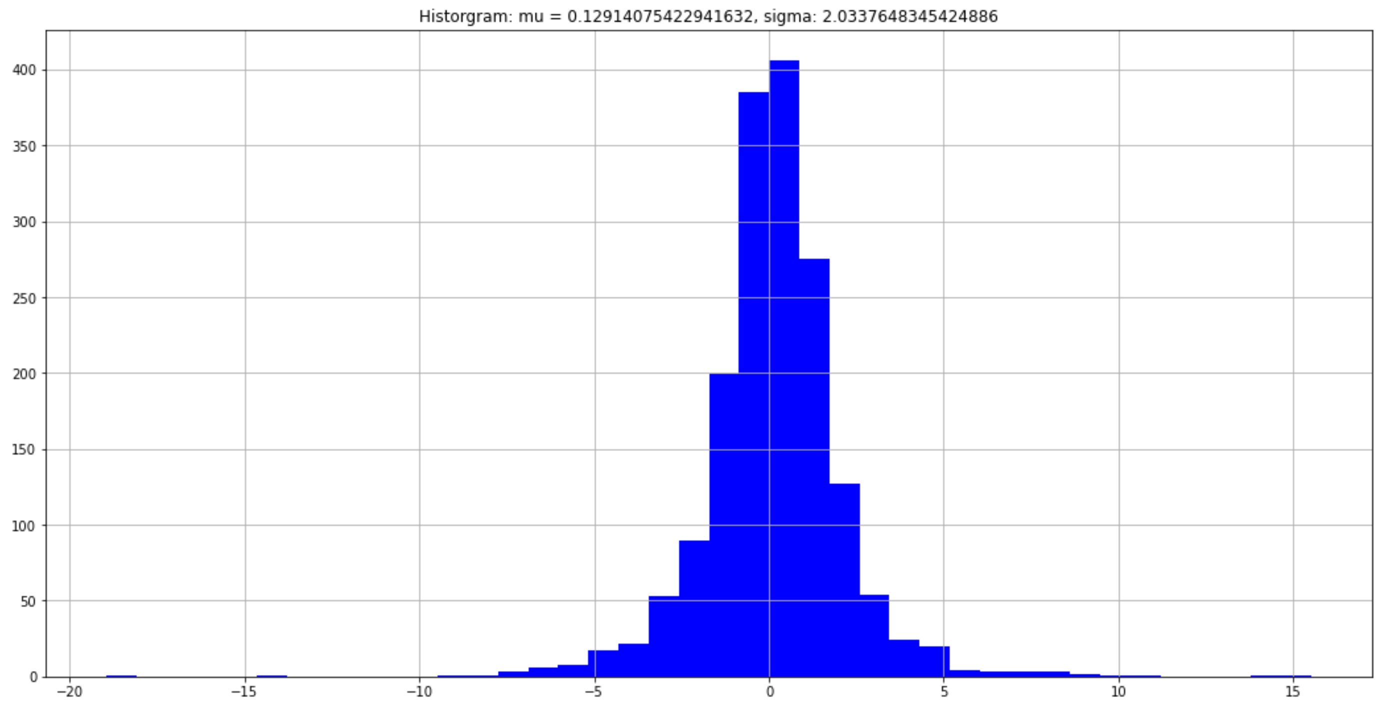



# 4. PLOT HISTOGRAMS

In [33]:
```
# 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 = (18,9))
plt.hist(daily_return_df['FB'], num_bins, facecolor = 'blue'); # ; is to get rid of ext
plt.grid()

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

Out[33]: Text(0.5, 1.0, 'Historgram: mu = 0.12914075422941632, sigma: 2.0337648345424886')

Loading [MathJax]/extensions/Safe.js

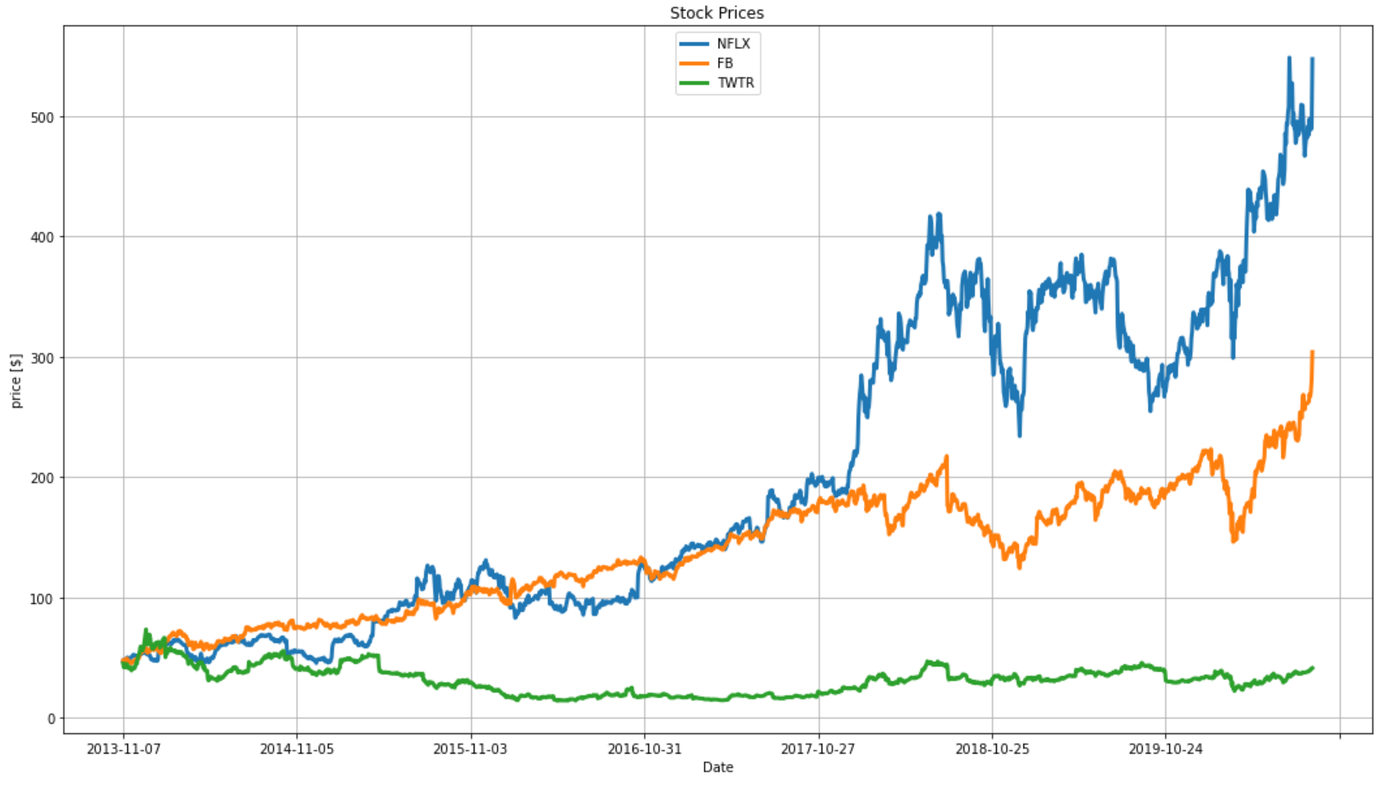



# 5. PLOT MULTIPLE PLOTS

In [34]:

```
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')
```

Out[34]: <matplotlib.legend.Legend at 0x18b6403ed00>

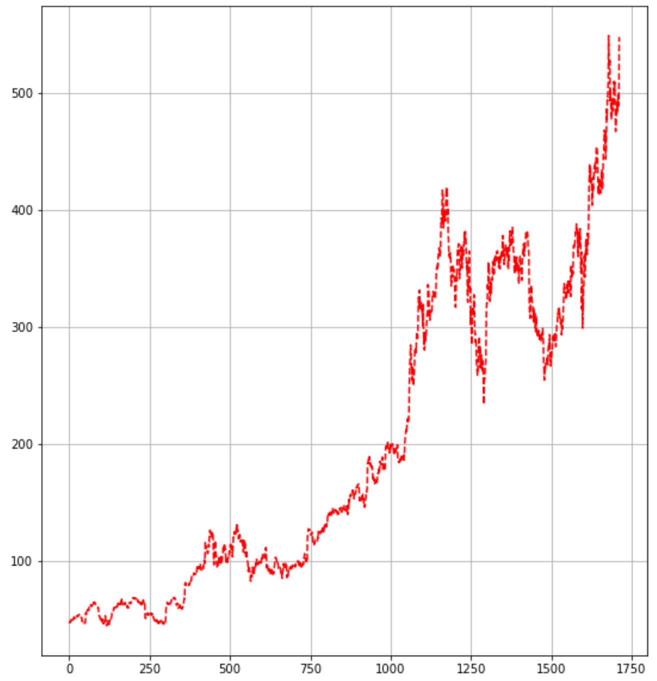


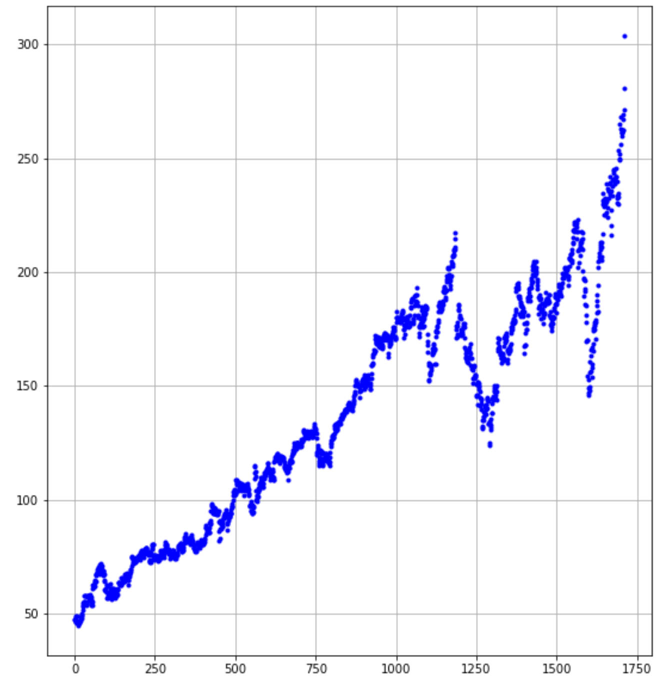

# 6. PLOT SUBPLOTS

In [35]:

```
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()
```

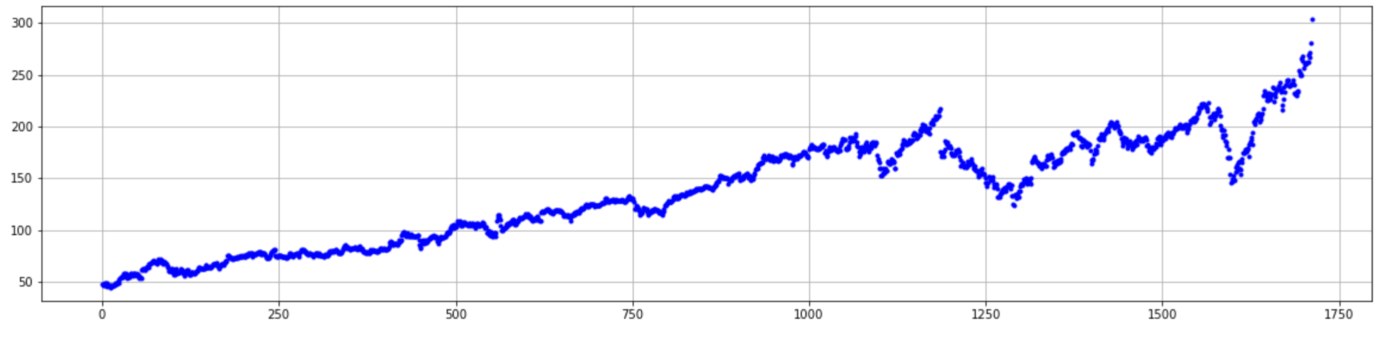
Loading [MathJax]/extensions/Safe.js




In [36]:

```
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()
```

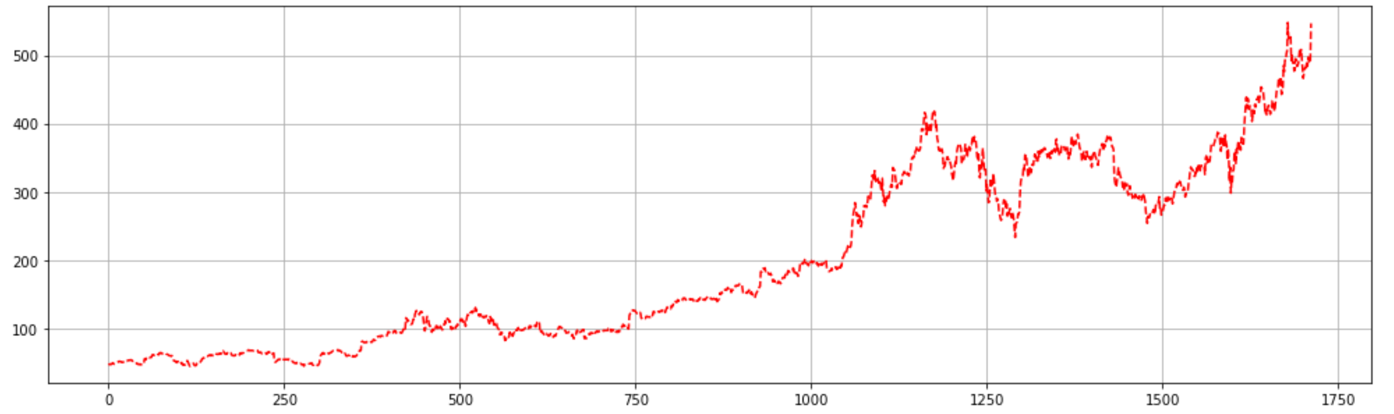



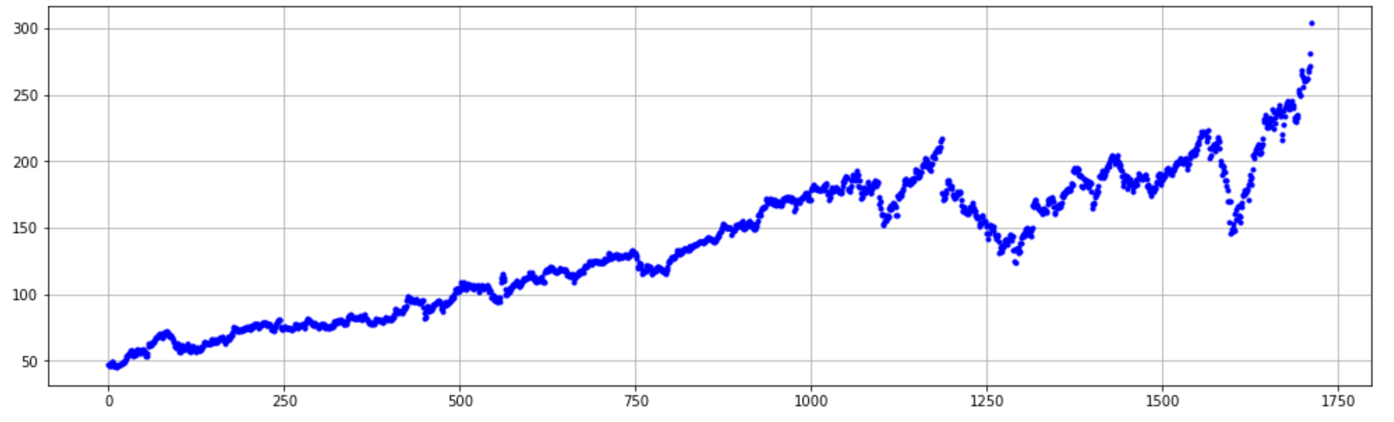

Explore more:

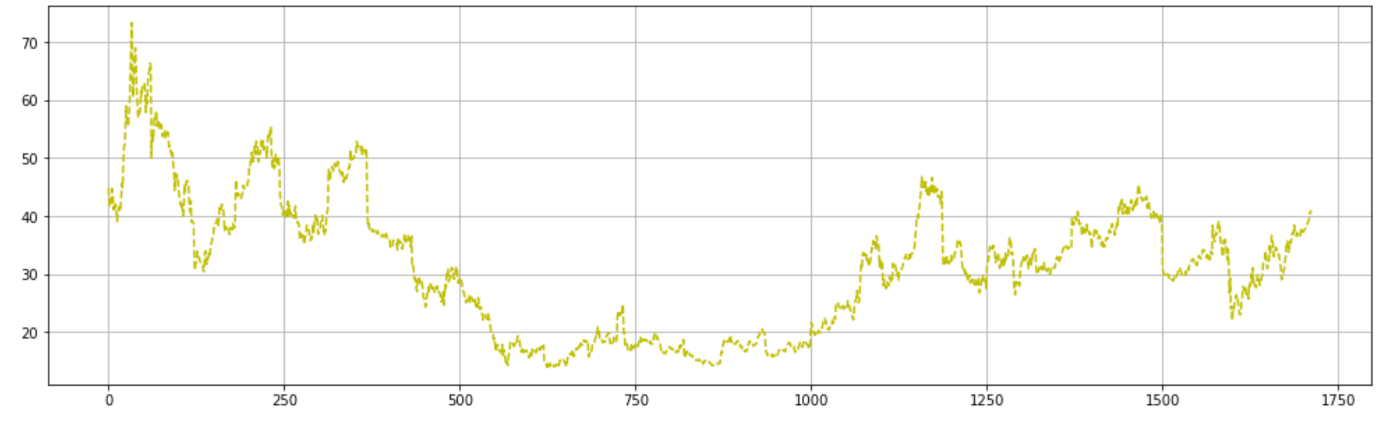
.Create subplots like above for Twitter, Facebook and Netflix

In [37]:

```
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()
```



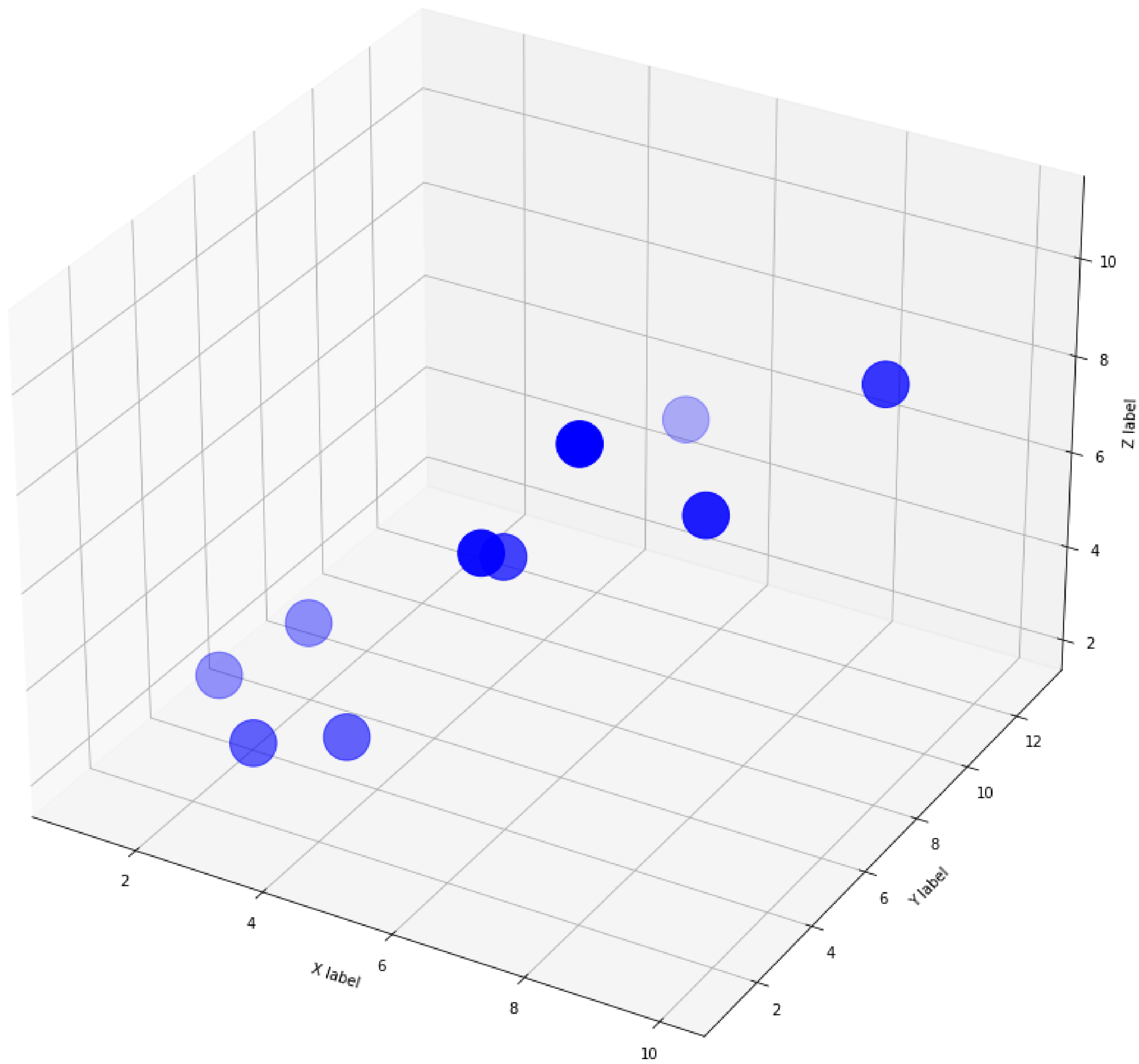

# 7. PLOT 3D PLOTS

In [41]:

```
# 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')
```

Out[41]:

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

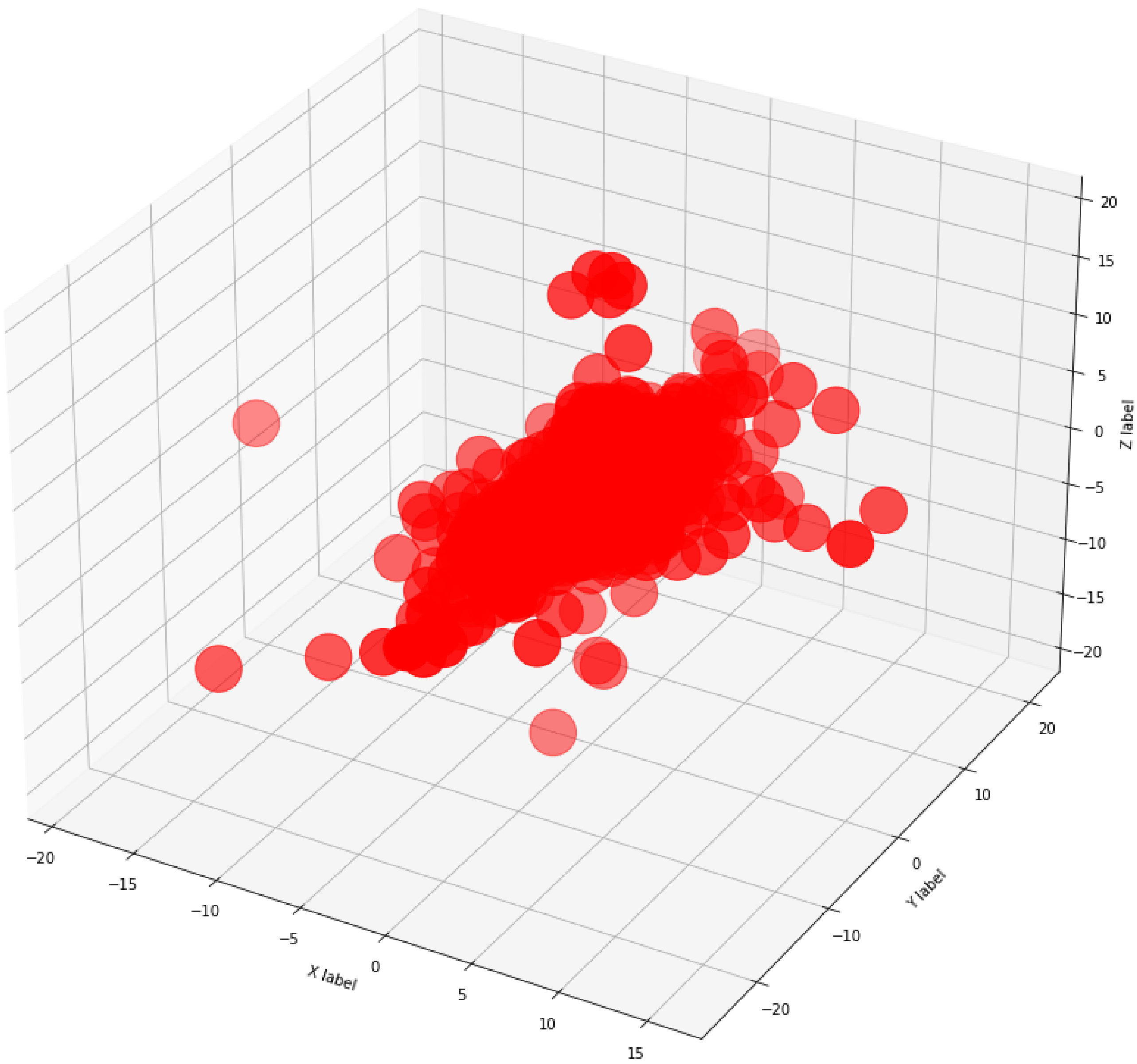



Explore more:

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

In [43]:
```
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')
```

Out[43]:
```
Text(0.5, 0, 'Z label')
```




# 8. SEABORN SCATTERPLOT & COUNTPLOT

In [44]:
```
# 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
```

In [45]:
```
# Import Cancer data drom the Sklearn library
from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()
cancer
```

Out[45]:
```
{'data': array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601e-01,
         1.189e-01],
        [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
         8.902e-02],
        [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
         8.758e-02],
        ...,
        [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
         7.820e-02],
        [2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
         1.240e-01],
        [7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
         7.039e-02]]),
 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
        0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
        1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
        1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
        1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
        0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
```

Loading [MathJax]/extensions/Safe.js

```
        1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
        1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
        0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
        1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
        1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0,
        0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
        0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
        1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
        1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
        1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
        1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
        1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
        1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1]),
 'frame': None,
 'target_names': array(['malignant', 'benign'], dtype='<U9'),
 'DESCR': '.. _breast_cancer_dataset:\n\nBreast cancer wisconsin (diagnostic) dataset\n
--------------------------------------------\n\n**Data Set Characteristics:**\n\n    :N
umber of Instances: 569\n\n    :Number of Attributes: 30 numeric, predictive attributes
and the class\n\n    :Attribute Information:\n        - radius (mean of distances from
center to points on the perimeter)\n        - texture (standard deviation of gray-scale
values)\n        - perimeter\n        - area\n        - smoothness (local variation in
radius lengths)\n        - compactness (perimeter^2 / area - 1.0)\n        - concavity
(severity of concave portions of the contour)\n        - concave points (number of conc
ave portions of the contour)\n        - symmetry\n        - fractal dimension ("coastli
ne approximation" - 1)\n\n        The mean, standard error, and "worst" or largest (mea
n of the three\n        worst/largest values) of these features were computed for each
image,\n        resulting in 30 features.  For instance, field 0 is Mean Radius, field
\n        10 is Radius SE, field 20 is Worst Radius.\n\n        - class:\n
- WDBC-Malignant\n                - WDBC-Benign\n\n    :Summary Statistics:\n\n    ====
================================= ====== ======\n
Min    Max\n    ===================================== ====== ======\n    radius (mean):
6.981  28.11\n    texture (mean):                       9.71   39.28\n    perimeter (me
an):                     43.79  188.5\n    area (mean):                          143.5
2501.0\n    smoothness (mean):                    0.053  0.163\n    compactness (mean):
0.019  0.345\n    concavity (mean):                     0.0    0.427\n    concave point
s (mean):                0.0    0.201\n    symmetry (mean):                      0.106
0.304\n    fractal dimension (mean):             0.05   0.097\n    radius (standard err
or):                0.112  2.873\n    texture (standard error):             0.36   4.885
\n    perimeter (standard error):           0.757  21.98\n    area (standard error):
6.802  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 e
rror):              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    :C
lass Distribution: 212 - Malignant, 357 - Benign\n\n    :Creator:  Dr. William H. Wolbe
rg, W. Nick Street, Olvi L. Mangasarian\n\n    :Donor: Nick Street\n\n    :Date: Novemb
er, 1995\n\nThis is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.\nht
tps://goo.gl/U2Uwz2\n\nFeatures are computed from a digitized image of a fine needle\na
spirate (FNA) of a breast mass.  They describe\ncharacteristics of the cell nuclei pres
ent in the image.\n\nSeparating plane described above was obtained using\nMultisurface
Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree\nConstruction Via Linear Programmin
g." Proceedings of the 4th\nMidwest Artificial Intelligence and Cognitive Science Socie
ty,\npp. 97-101, 1992], a classification method which uses linear\nprogramming to const
ruct a decision tree.  Relevant features\nwere selected using an exhaustive search in t
he 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 Lin
early Inseparable Sets",\nOptimization Methods and Software 1, 1992, 23-34].\n\nThis da
tabase 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 diagno
sis. IS&T/SPIE 1993 International Symposium on \n     Electronic Imaging: Science and T
echnology, volume 1905, pages 861-870,\n     San Jose, CA, 1993.\n   - O.L. Mangasaria
n, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and \n     prognosis via linea
r 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.',
Loading [MathJax]/extensions/Safe.js s': 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',
        'fractal dimension error', 'worst radius', 'worst texture',
        'worst perimeter', 'worst area', 'worst smoothness',
        'worst compactness', 'worst concavity', 'worst concave points',
        'worst symmetry', 'worst fractal dimension'], dtype='<U23'),
 'filename': 'C:\\Users\\lenovo\\anaconda3\\lib\\site-packages\\sklearn\\datasets\\data\\breast_cancer.csv'}
```

In [50]:

```
# Create a dataFrame named df_cancer with input/output data
df_cancer = pd.DataFrame(np.c_[cancer['data'], cancer['target']], columns = np.append(c
df_cancer.head(8)
```

Out[50]:

| | mean radius | mean texture | mean perimeter | mean area | mean smoothness | mean compactness | mean concavity | mean concave points | mean symmetry | mean fractal dimension | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 17.99 | 10.38 | 122.80 | 1001.0 | 0.11840 | 0.27760 | 0.30010 | 0.14710 | 0.2419 | 0.07871 | ... |
| **1** | 20.57 | 17.77 | 132.90 | 1326.0 | 0.08474 | 0.07864 | 0.08690 | 0.07017 | 0.1812 | 0.05667 | ... |
| **2** | 19.69 | 21.25 | 130.00 | 1203.0 | 0.10960 | 0.15990 | 0.19740 | 0.12790 | 0.2069 | 0.05999 | ... |
| **3** | 11.42 | 20.38 | 77.58 | 386.1 | 0.14250 | 0.28390 | 0.24140 | 0.10520 | 0.2597 | 0.09744 | ... |
| **4** | 20.29 | 14.34 | 135.10 | 1297.0 | 0.10030 | 0.13280 | 0.19800 | 0.10430 | 0.1809 | 0.05883 | ... |
| **5** | 12.45 | 15.70 | 82.57 | 477.1 | 0.12780 | 0.17000 | 0.15780 | 0.08089 | 0.2087 | 0.07613 | ... |
| **6** | 18.25 | 19.98 | 119.60 | 1040.0 | 0.09463 | 0.10900 | 0.11270 | 0.07400 | 0.1794 | 0.05742 | ... |
| **7** | 13.71 | 20.83 | 90.20 | 577.9 | 0.11890 | 0.16450 | 0.09366 | 0.05985 | 0.2196 | 0.07451 | ... |

8 rows × 31 columns

In [51]:

```
df_cancer.tail(8)
```

Out[51]:

| | mean radius | mean texture | mean perimeter | mean area | mean smoothness | mean compactness | mean concavity | mean concave points | mean symmetry | mean fractal dimension |
|---|---|---|---|---|---|---|---|---|---|---|
| **561** | 11.20 | 29.37 | 70.67 | 386.0 | 0.07449 | 0.03558 | 0.00000 | 0.00000 | 0.1060 | 0.05502 |
| **562** | 15.22 | 30.62 | 103.40 | 716.9 | 0.10480 | 0.20870 | 0.25500 | 0.09429 | 0.2128 | 0.07152 |
| **563** | 20.92 | 25.09 | 143.00 | 1347.0 | 0.10990 | 0.22360 | 0.31740 | 0.14740 | 0.2149 | 0.06879 |
| **564** | 21.56 | 22.39 | 142.00 | 1479.0 | 0.11100 | 0.11590 | 0.24390 | 0.13890 | 0.1726 | 0.05623 |
| **565** | 20.13 | 28.25 | 131.20 | 1261.0 | 0.09780 | 0.10340 | 0.14400 | 0.09791 | 0.1752 | 0.05533 |
| **566** | 16.60 | 28.08 | 108.30 | 858.1 | 0.08455 | 0.10230 | 0.09251 | 0.05302 | 0.1590 | 0.05648 |
| **567** | 20.60 | 29.33 | 140.10 | 1265.0 | 0.11780 | 0.27700 | 0.35140 | 0.15200 | 0.2397 | 0.07016 |
| **568** | 7.76 | 24.54 | 47.92 | 181.0 | 0.05263 | 0.04362 | 0.00000 | 0.00000 | 0.1587 | 0.05884 |

8 rows × 31 columns

In [55]:

```
# Plot scatter plot between mean area and mean smoothness
plt.figure(figsize = (15,10))
sns.scatterplot(x = 'mean area', y = 'mean smoothness', hue = 'target', data = df_cance
```

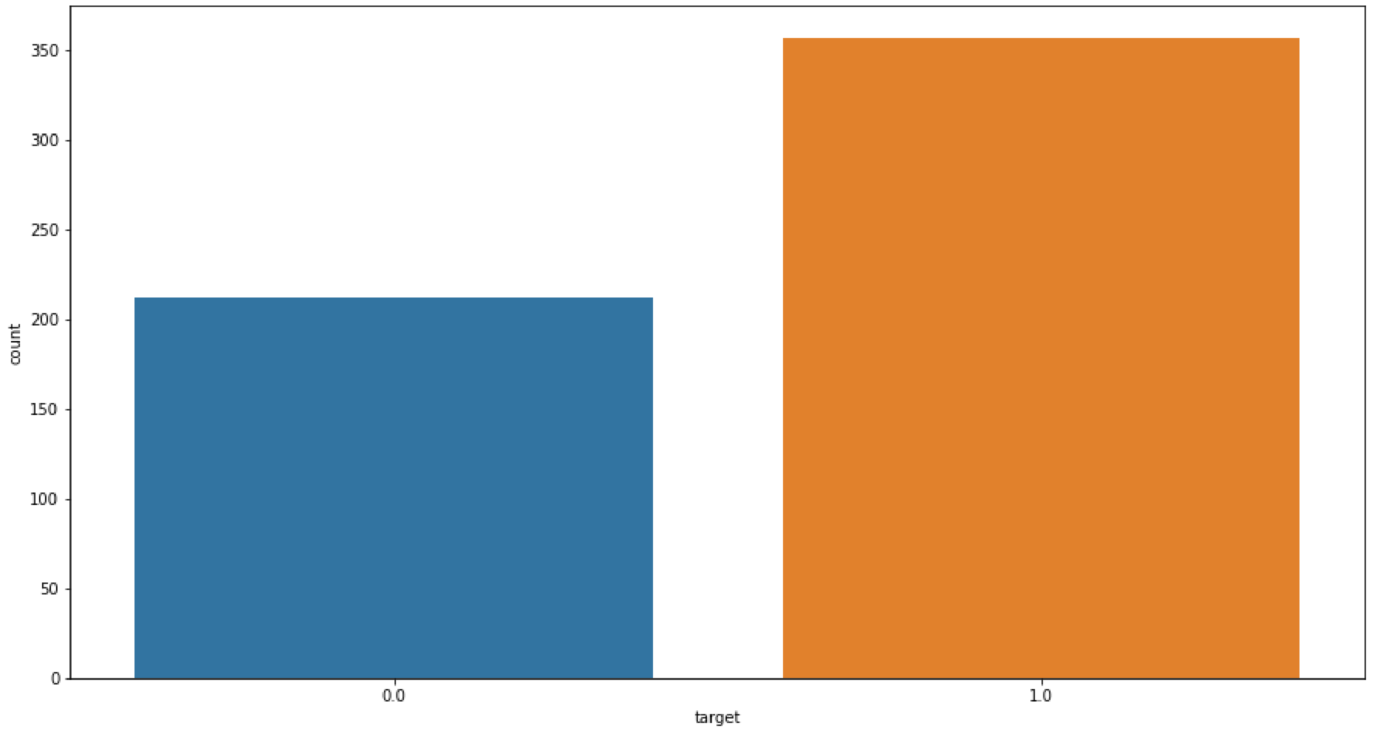
Out[55]:

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


In [57]:
```
# Let's print out countplot to know how many samples belong to class #0 and #1
plt.figure(figsize = (15,8))
sns.countplot(df_cancer['target'], label = 'Count')
```

```
C:\Users\lenovo\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: P
ass the following variable as a keyword arg: x. From version 0.12, the only valid posit
ional argument will be `data`, and passing other arguments without an explicit keyword
will result in an error or misinterpretation.
  warnings.warn(
```

Out[57]: <AxesSubplot:xlabel='target', ylabel='count'>

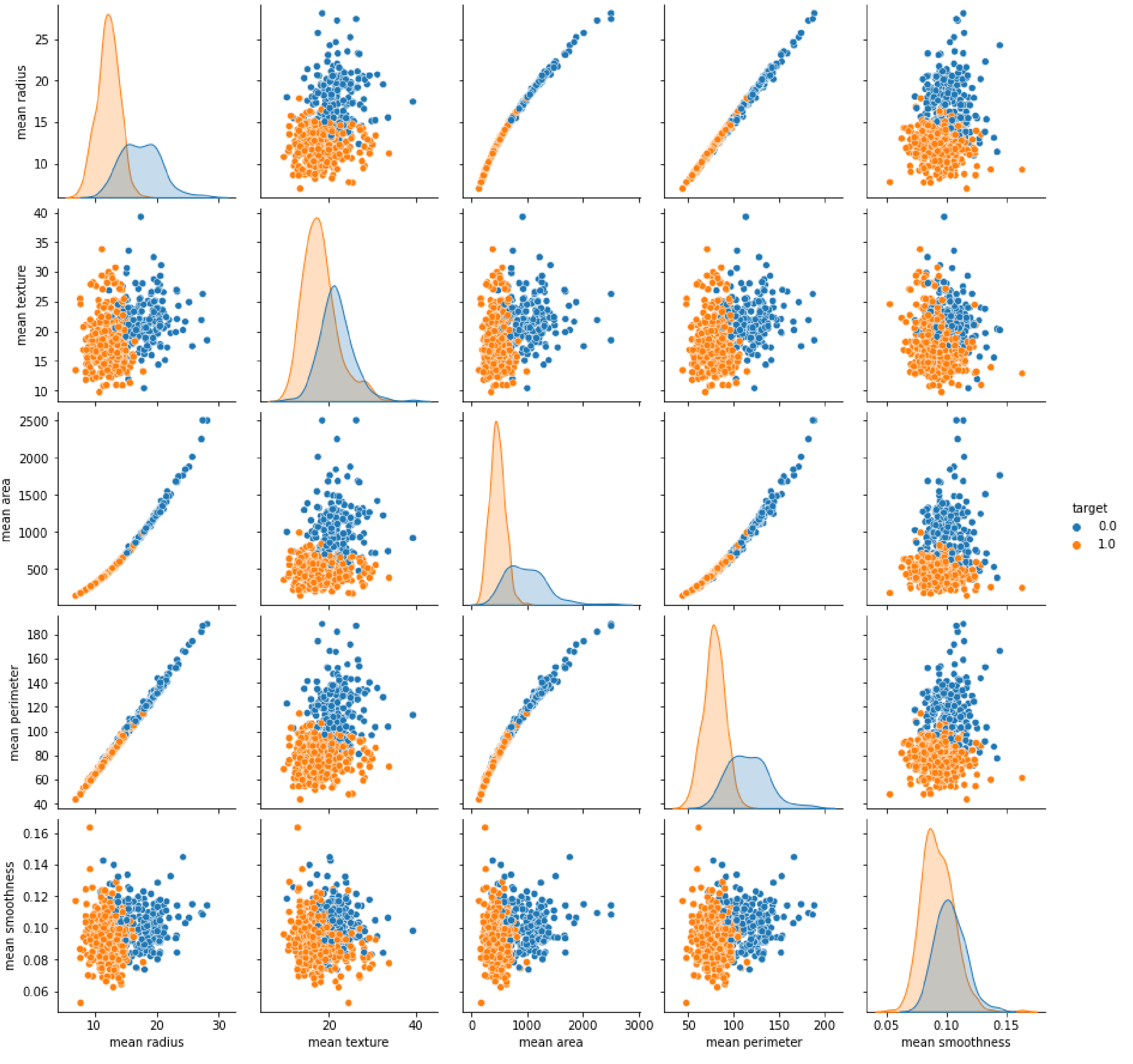



# 9. SEABORN PAIRPLOT, DISPLOT, AND HEATMAPS/CORRELATIONS

In [58]:
```
# Plot the pairplot
sns.pairplot(df_cancer, hue = 'target', vars = ['mean radius', 'mean texture', 'mean ar
```

Out[58]: <seaborn.axisgrid.PairGrid at 0x18b67d59e50>

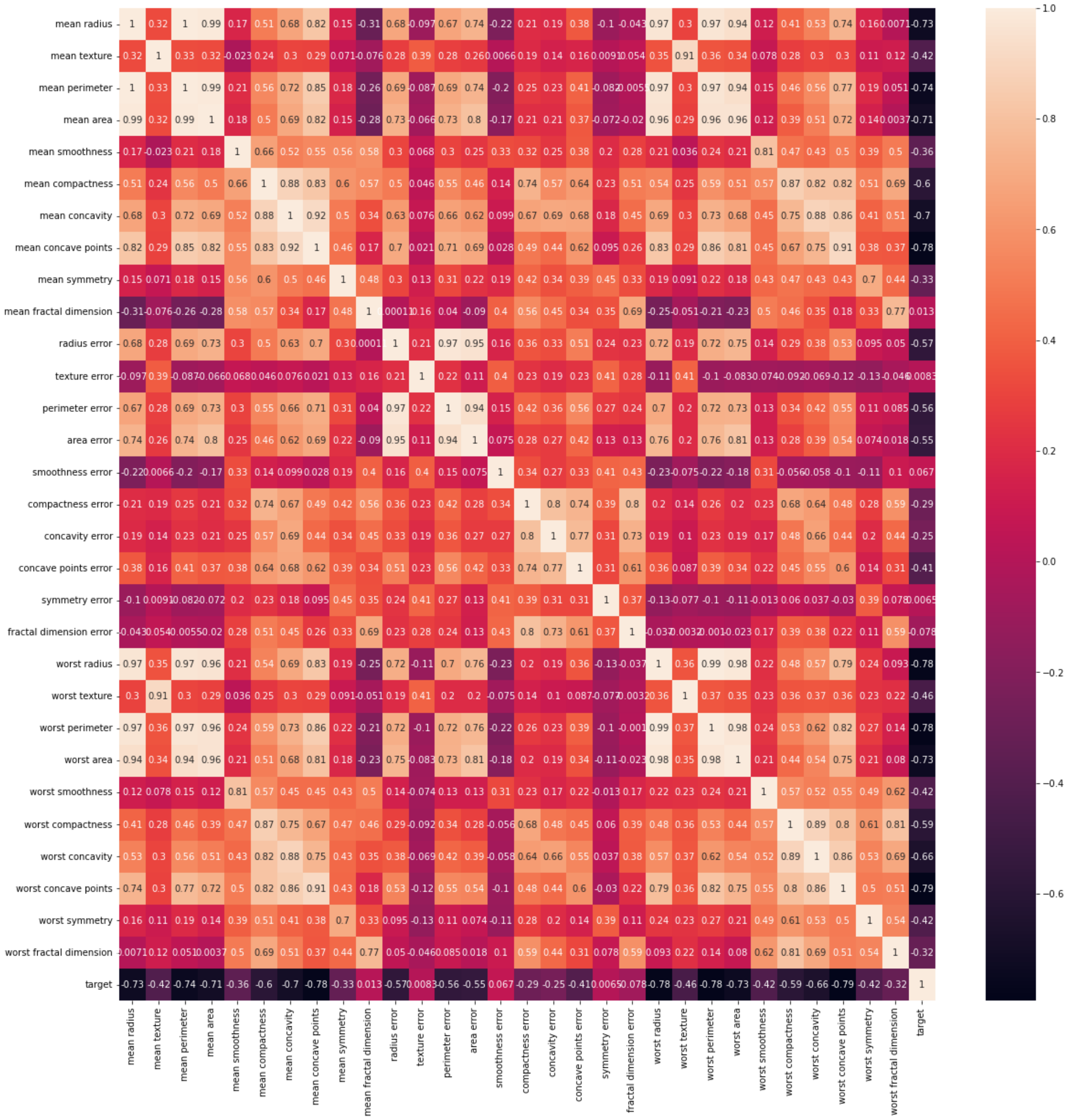
Loading [MathJax]/extensions/Safe.js




In [60]:
```
# Strong correlation between the mean radius and mean perimeter, mean area and mean pri
plt.figure(figsize = (20, 20))
sns.heatmap(df_cancer.corr(), annot = True)
```

Out[60]: <AxesSubplot:>

Loading [MathJax]/extensions/Safe.js

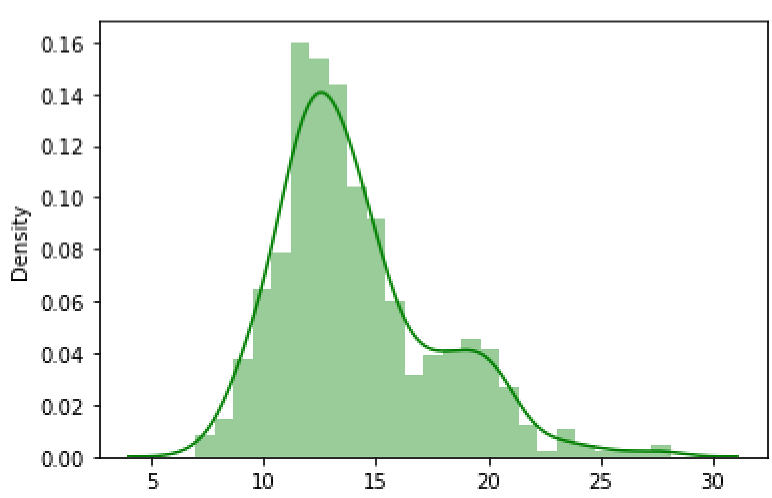



In [66]:

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

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

```
C:\Users\lenovo\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarnin
g: `distplot` is a deprecated function and will be removed in a future version. Please
adapt your code to use either `displot` (a figure-level function with similar flexibili
ty) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[66]: <AxesSubplot:xlabel='mean radius', ylabel='Density'>




Explore more:

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

Loading [MathJax]/extensions/Safe.js

In [67]:

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

In [68]:

```
class_0_df.head(10)
```

Out[68]:

| | mean radius | mean texture | mean perimeter | mean area | mean smoothness | mean compactness | mean concavity | mean concave points | mean symmetry | mean fractal dimension | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 17.99 | 10.38 | 122.80 | 1001.0 | 0.11840 | 0.27760 | 0.30010 | 0.14710 | 0.2419 | 0.07871 | ... |
| **1** | 20.57 | 17.77 | 132.90 | 1326.0 | 0.08474 | 0.07864 | 0.08690 | 0.07017 | 0.1812 | 0.05667 | ... |
| **2** | 19.69 | 21.25 | 130.00 | 1203.0 | 0.10960 | 0.15990 | 0.19740 | 0.12790 | 0.2069 | 0.05999 | ... |
| **3** | 11.42 | 20.38 | 77.58 | 386.1 | 0.14250 | 0.28390 | 0.24140 | 0.10520 | 0.2597 | 0.09744 | ... |
| **4** | 20.29 | 14.34 | 135.10 | 1297.0 | 0.10030 | 0.13280 | 0.19800 | 0.10430 | 0.1809 | 0.05883 | ... |
| **5** | 12.45 | 15.70 | 82.57 | 477.1 | 0.12780 | 0.17000 | 0.15780 | 0.08089 | 0.2087 | 0.07613 | ... |
| **6** | 18.25 | 19.98 | 119.60 | 1040.0 | 0.09463 | 0.10900 | 0.11270 | 0.07400 | 0.1794 | 0.05742 | ... |
| **7** | 13.71 | 20.83 | 90.20 | 577.9 | 0.11890 | 0.16450 | 0.09366 | 0.05985 | 0.2196 | 0.07451 | ... |
| **8** | 13.00 | 21.82 | 87.50 | 519.8 | 0.12730 | 0.19320 | 0.18590 | 0.09353 | 0.2350 | 0.07389 | ... |
| **9** | 12.46 | 24.04 | 83.97 | 475.9 | 0.11860 | 0.23960 | 0.22730 | 0.08543 | 0.2030 | 0.08243 | ... |

10 rows × 31 columns

In [69]:

```
class_1_df.head(10)
```

Out[69]:

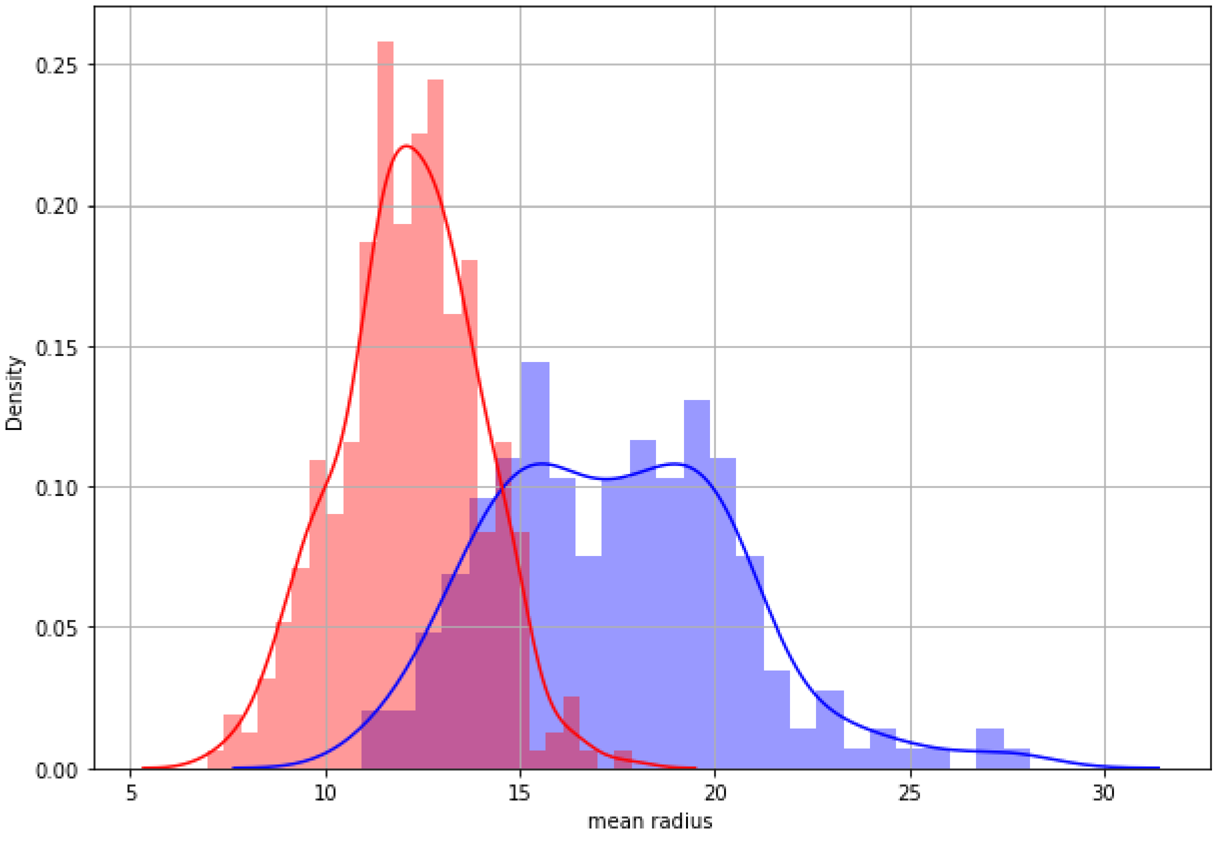
| | mean radius | mean texture | mean perimeter | mean area | mean smoothness | mean compactness | mean concavity | mean concave points | mean symmetry | mean fractal dimension | .. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **19** | 13.540 | 14.36 | 87.46 | 566.3 | 0.09779 | 0.08129 | 0.06664 | 0.047810 | 0.1885 | 0.05766 | .. |
| **20** | 13.080 | 15.71 | 85.63 | 520.0 | 0.10750 | 0.12700 | 0.04568 | 0.031100 | 0.1967 | 0.06811 | .. |
| **21** | 9.504 | 12.44 | 60.34 | 273.9 | 0.10240 | 0.06492 | 0.02956 | 0.020760 | 0.1815 | 0.06905 | .. |
| **37** | 13.030 | 18.42 | 82.61 | 523.8 | 0.08983 | 0.03766 | 0.02562 | 0.029230 | 0.1467 | 0.05863 | .. |
| **46** | 8.196 | 16.84 | 51.71 | 201.9 | 0.08600 | 0.05943 | 0.01588 | 0.005917 | 0.1769 | 0.06503 | .. |
| **48** | 12.050 | 14.63 | 78.04 | 449.3 | 0.10310 | 0.09092 | 0.06592 | 0.027490 | 0.1675 | 0.06043 | .. |
| **49** | 13.490 | 22.30 | 86.91 | 561.0 | 0.08752 | 0.07698 | 0.04751 | 0.033840 | 0.1809 | 0.05718 | .. |
| **50** | 11.760 | 21.60 | 74.72 | 427.9 | 0.08637 | 0.04966 | 0.01657 | 0.011150 | 0.1495 | 0.05888 | .. |
| **51** | 13.640 | 16.34 | 87.21 | 571.8 | 0.07685 | 0.06059 | 0.01857 | 0.017230 | 0.1353 | 0.05953 | .. |
| **52** | 11.940 | 18.24 | 75.71 | 437.6 | 0.08261 | 0.04751 | 0.01972 | 0.013490 | 0.1868 | 0.06110 | .. |

10 rows × 31 columns

In [70]:

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

```
C:\Users\lenovo\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarnin
g: `distplot` is a deprecated function and will be removed in a future version. Please
adapt your code to use either `displot` (a figure-level function with similar flexibili
ty) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\lenovo\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarnin
g: `distplot` is a deprecated function and will be removed in a future version. Please
adapt your code to use either `displot` (a figure-level function with similar flexibili
ty) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```


0.25
0.20
0.15
0.10
0.05
0.00
Density
5
10
15
20
25
30
mean radius