Principal Component Analysis Demo

Load Required Libraries

Some libraries come preinstalled in the Jupyter environment, others need to be installed.

In [1]:
```
import warnings
warnings.filterwarnings("ignore") # filter out warnings
import numpy as np
np.set_printoptions(formatter={'float': lambda x: "{0:0.4f}".format(x)}) # sets decima
np.set_printoptions(suppress=True) # disables scientific notation
import pandas as pd
import seaborn as sns
import plotly.express as px
import matplotlib.pyplot as plt
# %matplotlib inline
import pca
#pip install opencv-python
import cv2
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

In [2]:
```
from sklearn.datasets import load_breast_cancer
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

# Models
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression

# Model Performance Evaluators
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

Load Breast Cancer Dataset

Scikitlearn comes with a breast cancer dataset. Features in this dataset describe cell nuclei characteristics from digitized images of fine needle aspirate (FNA) of breast masses.

In [3]:
```
data = load_breast_cancer()
df = pd.DataFrame(data=data.data, columns=data.feature_names)
df['target'] = data.target
# Check out the dataset
df.head()
```

Out[3]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	dime
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0

5 rows × 31 columns

In [4]:
```
# Check out the dimensions of the dataset - dropping last target column
df[list(df.columns[:-1])].shape
```

Out[4]:
```
(569, 30)
```

Data Pre-processing

Standardize the dataset so that large ranges do not dominate the analysis

In [5]:
```
# Define input and target datasets
inputs_df = df[list(df.columns[:-1])]
targets = df[(df.columns[-1])]

# Standarize input dataset
sc = StandardScaler()
inputs_df[df.columns[:-1]] = sc.fit_transform(inputs_df[df.columns[:-1]])

df_scaled = inputs_df[df.columns[:-1]]
inputs_df[df.columns[:-1]].head()
```

Out[5]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	me symmet
0	1.097064	-2.073335	1.269934	0.984375	1.568466	3.283515	2.652874	2.532475	2.2175
1	1.829821	-0.353632	1.685955	1.908708	-0.826962	-0.487072	-0.023846	0.548144	0.0013
2	1.579888	0.456187	1.566503	1.558884	0.942210	1.052926	1.363478	2.037231	0.9396
3	-0.768909	0.253732	-0.592687	-0.764464	3.283553	3.402909	1.915897	1.451707	2.8673
4	1.750297	-1.151816	1.776573	1.826229	0.280372	0.539340	1.371011	1.428493	-0.0095

5 rows × 30 columns

In [6]:
```
# Produces a summary table
inputs_df[df.columns[:-1]].describe()
```

Out[6]:

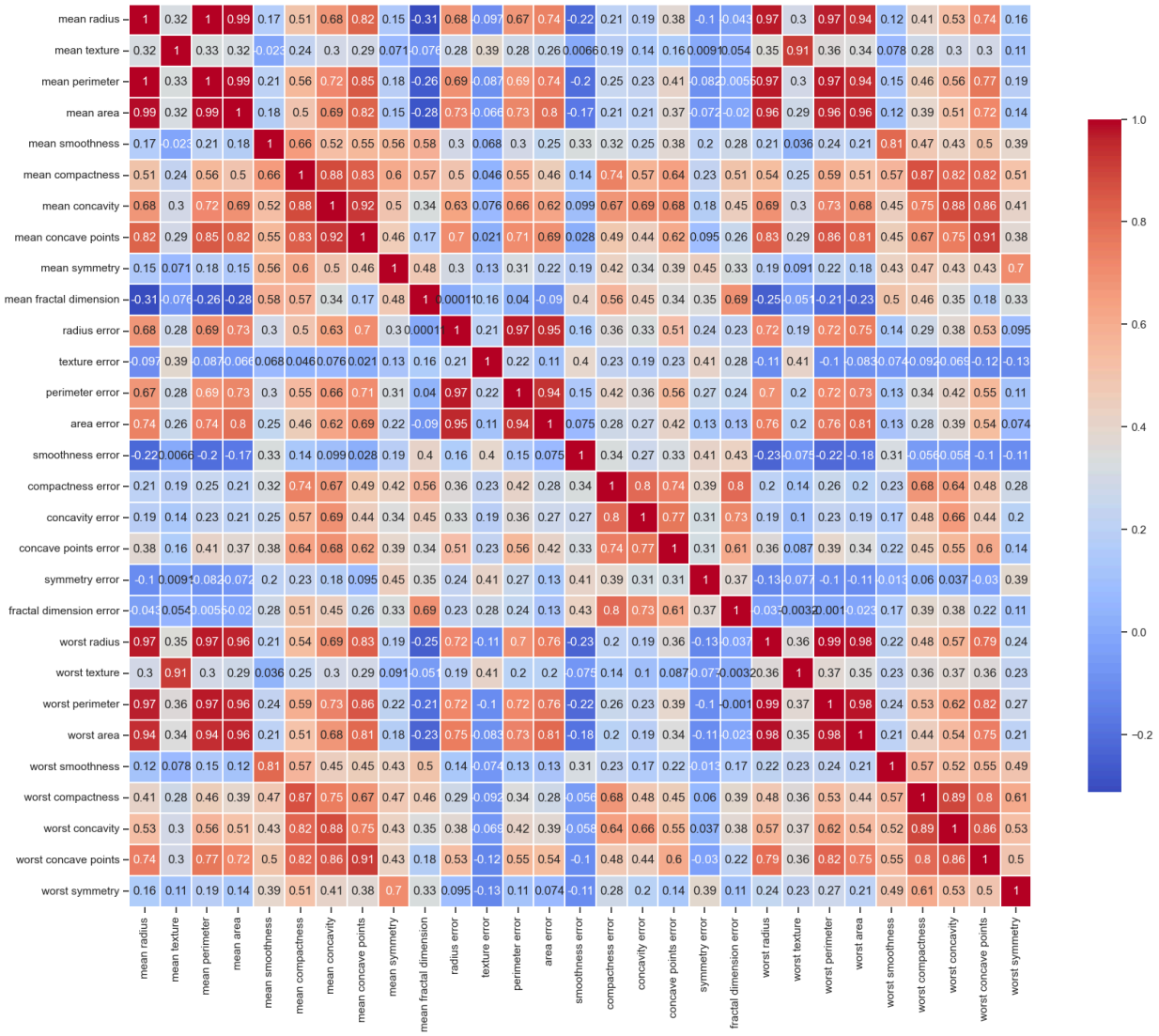
	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	
count	5.690000e+02	5.690000e+02	5.690000e+02	5.690000e+02	5.690000e+02	5.690000e+02	5.(
mean	-3.153111e-15	-6.568462e-15	-6.993039e-16	-8.553985e-16	6.081447e-15	-1.136369e-15	-2.
std	1.000880e+00	1.000880e+00	1.000880e+00	1.000880e+00	1.000880e+00	1.000880e+00	1.(
min	-2.029648e+00	-2.229249e+00	-1.984504e+00	-1.454443e+00	-3.112085e+00	-1.610136e+00	-1.
25%	-6.893853e-01	-7.259631e-01	-6.919555e-01	-6.671955e-01	-7.109628e-01	-7.470860e-01	-7.
50%	-2.150816e-01	-1.046362e-01	-2.359800e-01	-2.951869e-01	-3.489108e-02	-2.219405e-01	-3.
75%	4.693926e-01	5.841756e-01	4.996769e-01	3.635073e-01	6.361990e-01	4.938569e-01	5.
max	3.971288e+00	4.651889e+00	3.976130e+00	5.250529e+00	4.770911e+00	4.568425e+00	4.

8 rows × 30 columns

Principal Component Analysis

The heatmap will give us some insight into the relationships between features

In [7]:
```
# Heatmap of correlation coefficient matrix
sns.set(style='ticks', color_codes=True)
plt.rcParams['figure.figsize'] = [20,20]
sns.heatmap(inputs_df[list(inputs_df.columns[:-1])].astype(float).corr(), linewidths=0
            cbar_kws={'shrink': 0.6})
plt.show()
```



Calculate the covariance matrix and perform eigendecomposition

In [8]:

```
pca = PCA(n_components=30)
pca.fit(df_scaled)

print("Eigenvectors")
print(pca.components_)
print("Eigenvalues:")
print(pca.explained_variance_)
print()
print("Variances (Percentage):")
print(pca.explained_variance_ratio_ * 100)
print()
```

```
Eigenvectors
[[ 0.21890244  0.10372458  0.22753729  0.22099499  0.14258969  0.23928535
   0.25840048  0.26085376  0.13816696  0.06436335  0.20597878  0.01742803
   0.21132592  0.20286964  0.01453145  0.17039345  0.15358979  0.1834174
   0.04249842  0.10256832  0.22799663  0.10446933  0.23663968  0.22487053
   0.12795256  0.21009588  0.22876753  0.25088597  0.12290456  0.13178394]
 [-0.23385713 -0.05970609 -0.21518136 -0.23107671  0.18611302  0.15189161
   0.06016536 -0.0347675   0.19034877  0.36657547 -0.10555215  0.08997968
  -0.08945723 -0.15229263  0.20443045  0.2327159   0.19720728  0.13032156
   0.183848    0.28009203 -0.21986638 -0.0454673  -0.19987843 -0.21935186
   0.17230435  0.14359317  0.09796411 -0.00825724  0.14188335  0.27533947]
 [-0.00853124  0.0645499  -0.00931422  0.02869953 -0.1042919  -0.07409157
   0.00273384 -0.02556354 -0.04023994 -0.02257409  0.26848139  0.37463367
   0.26664537  0.21600653  0.30883898  0.15477972  0.17646374  0.22465757
   0.28858429  0.21150376 -0.04750699 -0.04229782 -0.04854651 -0.01190232
  -0.25979761 -0.23607563 -0.17305734 -0.17034408 -0.27131264 -0.23279131]
 [ 0.04140896 -0.60305     0.0419831   0.0534338   0.15938277  0.03179458
   0.01912275  0.06533594  0.06712498  0.04858676  0.09794124 -0.35985553
   0.08899241  0.10820504  0.04466418 -0.02746936  0.00131688  0.07406733
   0.04407335  0.01530475  0.01541724 -0.63280788  0.01380279  0.02589475
   0.01765222 -0.09132842 -0.07395118  0.006007   -0.03625069 -0.07705347]
 [ 0.03778635 -0.04946885  0.03737466  0.01033125 -0.36508853  0.01170397
   0.08637541 -0.04386103 -0.30594143 -0.04442436 -0.1544565  -0.19165051
  -0.12099022 -0.12757443 -0.23206568  0.27996816  0.35398209  0.19554809
  -0.25286876  0.26329744 -0.00440659 -0.0928834   0.00745415 -0.0273909
  -0.32443545  0.12180411  0.18851873  0.04333207 -0.24455866  0.09442335]
 [ 0.01874079 -0.03217884  0.01730844 -0.00188775 -0.2863745  -0.01413095
  -0.00934418 -0.05204995  0.35645846 -0.11943067 -0.02560326 -0.02874731
   0.00181072 -0.04286391 -0.34291739  0.06919752  0.05634324 -0.03122445
   0.49024564 -0.05319527 -0.00029068 -0.05000806  0.00850099 -0.02516438
  -0.36925537  0.04770579  0.02837926 -0.03087345  0.49892678 -0.08022352]
 [-0.12408834  0.01139954 -0.11447706 -0.05165343 -0.14066899  0.0309185
  -0.10752044 -0.15048221 -0.09389113  0.29576002  0.31249004 -0.09075536
   0.31464039  0.346679   -0.24402406  0.02346353 -0.20882379 -0.36964594
  -0.08038225  0.19139497 -0.00970994  0.00987074 -0.00044573  0.06783166
  -0.10883089  0.14047294 -0.06048806 -0.16796662 -0.01849063  0.37465763]
 [-0.0074523   0.13067483 -0.01868726  0.0346736  -0.28897458 -0.15139635
  -0.07282729 -0.15232241 -0.23153099 -0.17712144  0.02253997 -0.47541314
  -0.01189669  0.08580513  0.57341023  0.11746016  0.0605665  -0.10831931
   0.22014928  0.01116819  0.04261942  0.03625164  0.03055853  0.07939425
   0.20585219  0.08401966  0.07246787 -0.0361708   0.22822505  0.04836067]
 [-0.22310976  0.11269939 -0.22373921 -0.19558601  0.00642472 -0.16784142
   0.04059101 -0.11197111  0.25604008 -0.12374079  0.249985   -0.2466454
   0.22715402  0.22916002 -0.14192489 -0.14532281  0.35810708  0.27251989
  -0.3040772  -0.21372272 -0.11214146  0.1033412  -0.10961436 -0.08073246
   0.1123159  -0.10067782  0.16190862  0.06048846  0.06463781 -0.13417417]
 [ 0.09548644  0.24093407  0.08638562  0.07495649 -0.06929268  0.0129362
  -0.1356023   0.00805453  0.57206948  0.08110321 -0.04954759 -0.28914274
  -0.11450824 -0.09192789  0.16088461  0.04350487 -0.14127624  0.08624085
  -0.31652983  0.36754192  0.07736164  0.02955094  0.05050833  0.06992115
  -0.12830466 -0.17213363 -0.31163852 -0.07664829 -0.02956308  0.01260958]
 [ 0.04147149 -0.3022434   0.01678264  0.11016964 -0.13702184 -0.30800963
   0.12419024 -0.07244603  0.16305408 -0.03804827 -0.02535702  0.34494446
  -0.16731877  0.05161946  0.08420621 -0.20688568  0.34951794 -0.34237591
  -0.18784404  0.25062479  0.10506733  0.01315727  0.05107628  0.18459894
   0.14389035 -0.19742047  0.18501676 -0.11777205  0.15756025  0.11828355]
 [ 0.05106746  0.25489642  0.03892611  0.06543751  0.31672721 -0.10401704
   0.06565348  0.04258927 -0.2888655   0.23635899 -0.01668792 -0.30616042
  -0.10144683 -0.01767922 -0.29471005 -0.26345651  0.25114697 -0.00645875
   0.32057135  0.27616597  0.03967967  0.07979745 -0.00898774  0.04808866
```

```
  0.05651487 -0.3716625  -0.08703453 -0.06812535  0.0440335  -0.03473169]
[ 0.01196721  0.20346133  0.0441095   0.06737574  0.0455736   0.2292813
  0.38709081  0.1321381   0.18993367  0.10623908 -0.06819523 -0.16822238
 -0.03784399  0.05606493  0.15044143  0.01004017  0.15878319 -0.49402674
  0.01033274 -0.24045832 -0.13789053 -0.08014543 -0.09696571 -0.10116061
 -0.20513034  0.01227931  0.21798433 -0.25438749 -0.25653491 -0.17281424]
[ 0.05950613 -0.0215601   0.04851381  0.01083083  0.44506486  0.00810106
 -0.1893587  -0.24479477  0.03073886 -0.37707887  0.01034741 -0.01084935
 -0.04552372  0.08357072 -0.20115253  0.49175593  0.13458692 -0.19966672
 -0.04686438  0.14565247  0.02310128  0.05343079  0.01221938 -0.00668546
  0.16223544  0.16647025 -0.06679893 -0.27641889  0.00535557 -0.21210411]
[ 0.05111877  0.10792242  0.03990294 -0.01396691  0.11814336 -0.23089996
  0.12828373  0.21709919  0.07396171 -0.51797571  0.11005071 -0.03275272
  0.00826809  0.04602437 -0.01855946 -0.16820932 -0.25047141 -0.06207934
  0.1133832   0.35323221 -0.16656707 -0.1011154  -0.1827552  -0.3149936
 -0.04612587  0.04995601  0.20483589  0.16949961 -0.13988839  0.2561732 ]
[-0.15058388 -0.15784196 -0.11445396 -0.13244803 -0.20461325  0.17017837
  0.26947021  0.3804641  -0.16466159 -0.04079279  0.05890572 -0.0345004
  0.02651665  0.04115323 -0.05803906  0.1898309  -0.12542065 -0.19881035
 -0.1577115   0.26855388 -0.08156057  0.18555785 -0.05485705 -0.09065339
  0.14555166 -0.15373486 -0.21502195  0.17814174  0.25789401 -0.40555649]
[ 0.20292425 -0.03870612  0.19482131  0.25570576  0.16792991 -0.02030771
 -0.00159835  0.03450951 -0.19173785  0.05022525 -0.13939687  0.04396302
 -0.02463564  0.33441817  0.13959501 -0.00824648  0.08461672  0.10813226
 -0.27405913 -0.1227334  -0.24004998  0.06936519 -0.23416415 -0.27339958
 -0.2780302  -0.00403712 -0.19131342 -0.07548532  0.43065812  0.1593943 ]
[ 0.14671234 -0.04110299  0.15831745  0.2661681  -0.3522268   0.00779414
 -0.02696811 -0.08282774  0.17339779  0.08786736 -0.23621653 -0.00985866
 -0.0259288   0.3049069  -0.23125994  0.10047423 -0.00019549  0.04605491
  0.18701476 -0.0598231  -0.21610135  0.05839845 -0.18854359 -0.14206486
  0.50155168 -0.07357451 -0.10390798  0.0758139  -0.27871384  0.02356475]
[-0.22538466 -0.02978864 -0.23959528  0.02732219  0.16456584 -0.28422236
 -0.00226636  0.15497236  0.05881116  0.05815705 -0.17588331 -0.03600985
 -0.36570154  0.41657231  0.01326009  0.24244818 -0.12638102  0.0121643
  0.08903929 -0.08660084 -0.0136613   0.07586693 -0.09081325  0.4100472
 -0.23451384 -0.0202007   0.04578612  0.26022962 -0.11725053  0.01149448]
[-0.04969866 -0.24413499 -0.01766501 -0.09014376  0.01710096  0.48868633
 -0.03338709 -0.23540761  0.02606916 -0.17563722 -0.0908005  -0.07165999
 -0.17725063  0.27420115  0.09006148 -0.46109822  0.06694617  0.06886829
  0.10738529  0.2223453  -0.00562691  0.3005998   0.01100386  0.06004739
 -0.1297239   0.22928059 -0.04648279  0.03302234 -0.11675924 -0.10499197]
[-0.06857001  0.44836947 -0.06976904 -0.01844328 -0.11949175  0.1926214
  0.00557175 -0.00942382 -0.08693848 -0.07627184  0.08638677  0.21707197
 -0.30495016  0.19258779 -0.07209873 -0.14038657  0.06304793  0.03437532
 -0.09769953  0.06284328  0.0072939  -0.59444014 -0.0920236   0.14679013
  0.16484924  0.18137487 -0.13210059  0.00088608  0.16270855 -0.09234394]
[ 0.0729289   0.09480063  0.07516048  0.09756578  0.06382295 -0.09807756
 -0.185212   -0.31185243 -0.01840673  0.28786888 -0.15027468  0.04845693
  0.1593528   0.06423262  0.0505449  -0.04528769 -0.20521269 -0.07254538
 -0.08465443  0.24470508 -0.09629821 -0.11111202  0.01722163 -0.09695982
 -0.06825409  0.02967641  0.46042619  0.29984056  0.09714484 -0.46947115]
[-0.09855269 -0.000555   -0.04024471  0.00777727 -0.02066572  0.0523604
  0.32487038 -0.0514088  -0.05120058 -0.08468986 -0.26412532 -0.00087388
  0.09007421  0.09821507 -0.05981772  0.00910387 -0.38754233  0.35175507
 -0.04236289  0.0857811  -0.05567679 -0.0089229   0.06334483  0.19088963
  0.09369015 -0.14792092  0.28643314 -0.5675278   0.12134345  0.00762534]
[ 0.18257944 -0.09878679  0.11664888 -0.06984834 -0.06869742  0.10413552
 -0.04474106 -0.0840277  -0.01933947  0.13326055  0.55870157 -0.0242673
 -0.51675039  0.02246072 -0.01563119  0.12177779 -0.18820504  0.10966898
 -0.0032262  -0.07519442  0.15683037  0.1184846  -0.23711317 -0.14406303
```

```
   0.01099014 -0.18674995  0.28885257 -0.10734024  0.01438181 -0.03782545]
 [ 0.0192265  -0.08474593 -0.02701541  0.21004078 -0.02895489 -0.39662323
   0.09697732  0.1864516   0.02458369  0.20722186  0.17493043 -0.05698648
  -0.07292764 -0.13185041 -0.0312107  -0.17316455 -0.01593998  0.12954655
   0.01951493  0.0841712  -0.07070972  0.11818972 -0.11803403  0.03828995
   0.04796476  0.62438494 -0.11577034 -0.26319634 -0.04529962 -0.28013348]
 [-0.1294764  -0.02455666 -0.12525595  0.3627274  -0.03700369  0.26280847
  -0.54887617  0.38764338 -0.01604404 -0.09740484  0.04997708 -0.01123724
   0.10365328 -0.15530459 -0.00771756 -0.04972763  0.09145497 -0.01794192
  -0.01726785  0.03548897 -0.19705474  0.03646943 -0.24410367  0.23135953
   0.01260246 -0.10046342  0.26685378 -0.13357451  0.0281843   0.00452048]
 [-0.13152667 -0.01735731 -0.11541542  0.46661248  0.06968992  0.09774871
   0.3648084  -0.45469935 -0.01516483 -0.10124495  0.2129829  -0.01009289
   0.04169155 -0.31335866 -0.00905215  0.04653609 -0.0842248  -0.01116551
  -0.01997598 -0.01203656 -0.17866674  0.02141069 -0.24103105  0.23716247
  -0.04085357 -0.07050541 -0.1429058   0.23090139  0.02279044  0.059986  ]
 [ 0.21119401 -0.00006581  0.08433827 -0.27250832  0.00147927 -0.00546277
   0.04553864 -0.0088831   0.00143303 -0.00631169 -0.19222389 -0.00562261
   0.26319187 -0.04206811  0.00979296 -0.01539555  0.00582098 -0.0290093
  -0.00763653  0.01975646  0.41263958 -0.00039025 -0.7286809   0.23896032
  -0.00153525  0.04869182 -0.0176409   0.02247567  0.00492048 -0.02356214]
 [ 0.21146046 -0.01053393  0.3838261  -0.42279492 -0.00343467 -0.04101677
  -0.01001479 -0.00420695 -0.00756986  0.00730143  0.11844211 -0.00877628
  -0.00610022 -0.08592591  0.00177639  0.00315813  0.01607852 -0.02393779
  -0.00522329 -0.00834191 -0.63572492  0.01723549  0.0229218   0.44493593
   0.00738549  0.00000357 -0.01267572  0.03524045  0.01340423  0.01147766]
 [-0.70241409 -0.00027366  0.68989697  0.03294735  0.00484746 -0.04467419
  -0.02513867  0.00107727  0.00128038  0.00475568  0.00871109  0.00107104
  -0.01372939 -0.00110533  0.00160821 -0.00191562  0.00892653  0.0021602
  -0.00032939 -0.00179896  0.13564306 -0.00102054 -0.07974385 -0.03974228
  -0.00458328  0.01284156 -0.00040214  0.00228844 -0.00039544 -0.00189429]]
Eigenvalues:
[13.30499079  5.7013746   2.82291016  1.98412752  1.65163324  1.20948224
  0.67640888  0.47745625  0.41762878  0.35131087  0.29443315  0.26162116
  0.24178242  0.15728615  0.0943007   0.0800034   0.05950361  0.05271142
  0.0495647   0.03121426  0.03002566  0.02748771  0.02438369  0.01808679
  0.01550853  0.00819204  0.00691261  0.00159214  0.00075012  0.00013328]

Variances (Percentage):
[44.27202561 18.97118204  9.39316326  6.60213492  5.49576849  4.02452204
  2.25073371  1.5887238   1.38964937  1.16897819  0.97971899  0.8705379
  0.80452499  0.52336575  0.31378322  0.26620934  0.19799679  0.17539595
  0.16492531  0.10386467  0.09990965  0.09146468  0.08113613  0.06018336
  0.05160424  0.0272588   0.02300155  0.00529779  0.00249601  0.00044348]
```

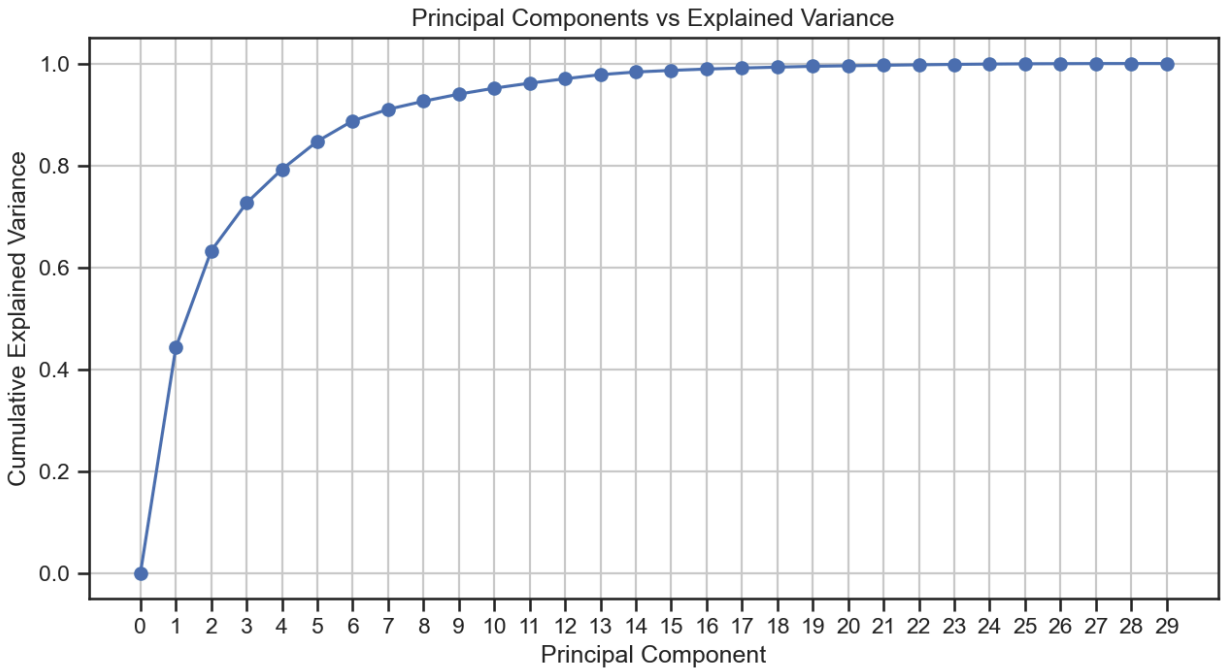
In [9]:

```
# Creates a plot of eigenvectors and the cumulative variance explained
nums = np.arange(30)
var_ratio = []
for num in nums:
  pca = PCA(n_components=num)
  pca.fit(df_scaled)
  var_ratio.append(np.sum(pca.explained_variance_ratio_))

plt.figure(figsize=(10,5),dpi=150)
plt.grid()
plt.plot(nums,var_ratio,marker='o')
plt.xticks(np.arange(0,30,step=1))
plt.xlabel('Principal Component')
```

```
plt.ylabel('Cumulative Explained Variance')
plt.title('Principal Components vs Explained Variance')
```

Out[9]:
```
Text(0.5, 1.0, 'Principal Components vs Explained Variance')
```



In [10]:
```
# Transform data using 6 PCs
pc = pca.components_[0:6]
transformed_df = np.dot(df.iloc[:,0:30],pc.T)
new_df = pd.DataFrame(transformed_df,columns=['PC1','PC2','PC3', 'PC4', 'PC5', 'PC6'])
new_df['Target'] = df['target'].values
new_df['Target'] = new_df['Target'].astype('int')
new_df.head()
```

Out[10]:

	PC1	PC2	PC3	PC4	PC5	PC6	Target
0	793.364674	-772.386400	28.693570	114.456637	-61.498704	-56.235935	0
1	831.376304	-819.573038	21.695304	113.079259	-46.296461	-52.514218	0
2	751.111800	-737.924156	26.218798	98.643829	-44.035575	-47.209793	0
3	271.973741	-262.394926	4.912013	14.387632	-15.271273	-15.455116	0
4	741.248858	-730.596479	30.886047	110.336027	-38.163621	-43.346078	0

Running a Prediction Model

In [11]:
```
#Splitting the data into train and validation set
train_inputs, val_inputs, train_targets, val_targets = train_test_split(new_df, target
```

In [12]:
```
names = ['Logistic Regression', "KNN", "Linear SVM","Gradient Boosting", "Decision Tre
classifiers = [
    LogisticRegression(solver='liblinear', random_state=42),
    KNeighborsClassifier(n_neighbors=3, weights= 'distance'),
    SVC(kernel="linear", C=0.025,  random_state=42),
    GradientBoostingClassifier(n_estimators=100, random_state=42),
```

```
    DecisionTreeClassifier(max_depth=5, random_state=42),
    RandomForestClassifier(max_depth=5, n_estimators=100, random_state=42)]
```

In [13]:
```
scores = []
for name, clf in zip(names, classifiers):
    clf.fit(train_inputs, train_targets)
    score = clf.score(val_inputs, val_targets)
    scores.append(score)
scores_df = pd.DataFrame()
scores_df['name'] = names
scores_df['score'] = np.around(scores,3)
scores_df.sort_values('score', ascending= False)
```

Out[13]:

	name	score
0	Logistic Regression	1.000
2	Linear SVM	1.000
3	Gradient Boosting	1.000
4	Decision Tree	1.000
5	Random_Forest	1.000
1	KNN	0.937

In [14]:
```
# Load dataset
data = load_breast_cancer()
X = data.data
y = data.target
col_labels = data.feature_names

# Load library
from pca import pca

# Initialize pca with default parameters
model = pca(normalize=True)

# Fit transform and include the column labels and row labels
results = model.fit_transform(X, col_labels=col_labels, row_labels=y)

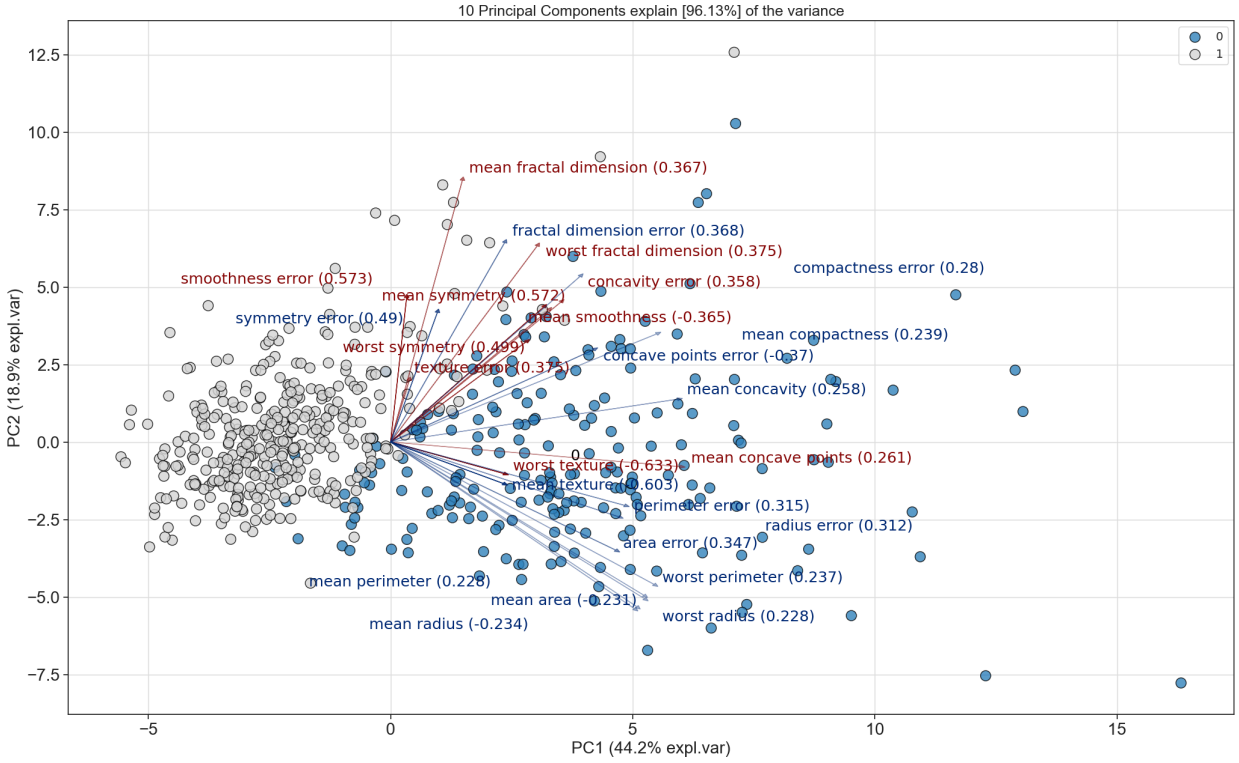
# Scatter plot with loadings
model.biplot()
```

```
[scatterd] >INFO> Create scatterplot
```

```
[pca] >Normalizing input data per feature (zero mean and unit variance)..
[pca] >The PCA reduction is performed to capture [95.0%] explained variance using the
[30] columns of the input data.
[pca] >Fit using PCA.
[pca] >Compute loadings and PCs.
[pca] >Compute explained variance.
[pca] >Number of components is [10] that covers the [95.00%] explained variance.
[pca] >The PCA reduction is performed on the [30] columns of the input dataframe.
[pca] >Fit using PCA.
[pca] >Compute loadings and PCs.
[pca] >Outlier detection using Hotelling T2 test with alpha=[0.05] and n_components=
[10]
[pca] >Multiple test correction applied for Hotelling T2 test: [fdr_bh]
[pca] >Outlier detection using SPE/DmodX with n_std=[3]
[pca] >Plot PC1 vs PC2 with loadings.
```

Out[14]:

```
(<Figure size 2500x1500 with 1 Axes>,
 <Axes: title={'center': '10 Principal Components explain [96.13%] of the variance'},
xlabel='PC1 (44.2% expl.var)', ylabel='PC2 (18.9% expl.var)'>)
```



Bonus: Image Compression using PCA

In [15]:

```
# Loading the image
img_load = cv2.imread('zeus.jpg')
img = cv2.cvtColor(img_load, cv2.COLOR_BGR2RGB)
plt.imshow(img)
```

Out[15]:

```
<matplotlib.image.AxesImage at 0x1e113c10910>
```



In [16]:
```
img.shape
```

Out[16]:
```
(707, 750, 3)
```

In [17]:
```
print(img)
```

```
[[[157 157 159]
  [157 157 159]
  [157 157 159]
  ...
  [134 129 125]
  [134 129 125]
  [135 130 126]]

 [[157 157 159]
  [157 157 159]
  [157 157 159]
  ...
  [135 130 126]
  [135 130 126]
  [135 130 126]]

 [[158 158 160]
  [158 158 160]
  [158 158 160]
  ...
  [135 130 126]
  [135 130 126]
  [135 130 126]]

 ...

 [[ 62  63  67]
  [ 61  62  66]
  [ 61  62  66]
  ...
  [126 114  92]
  [123 111  89]
  [122 110  88]]

 [[ 62  63  67]
  [ 62  63  67]
  [ 61  62  66]
  ...
  [126 114  92]
  [124 112  90]
  [122 110  88]]

 [[ 63  64  68]
  [ 63  64  68]
  [ 61  62  66]
  ...
  [127 115  93]
  [124 112  90]
  [123 111  89]]]
```

In [18]:

```
# Splitting the image in R,G,B arrays.

red,green,blue = cv2.split(img)
# it will split the original image into Blue, Green and Red arrays.

# initialize PCA with first 20 principal components
# change components to 100
pca = PCA(20)

# Applying to red channel and then applying inverse transform to transformed array.
```

```
red_transformed = pca.fit_transform(red)
red_inverted = pca.inverse_transform(red_transformed)

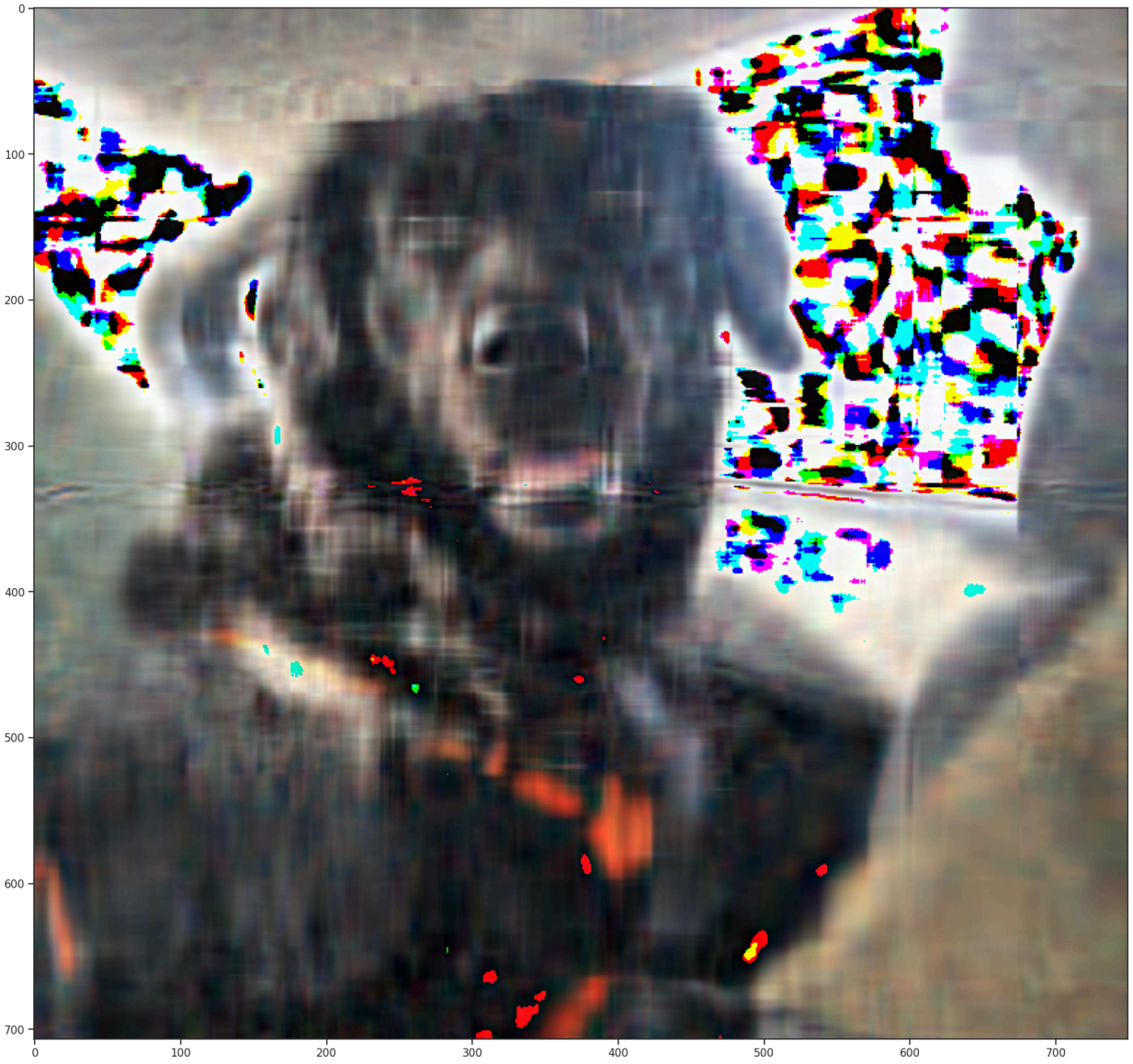
#Applying to Green channel and then applying inverse transform to transformed array.
green_transformed = pca.fit_transform(green)
green_inverted = pca.inverse_transform(green_transformed)

#Applying to Blue channel and then applying inverse transform to transformed array.
blue_transformed = pca.fit_transform(blue)
blue_inverted = pca.inverse_transform(blue_transformed)

img_compressed = (np.dstack((red_inverted, green_inverted, blue_inverted))).astype(np.
#viewing the compressed image
plt.imshow(img_compressed)
```

Out[18]:

```
<matplotlib.image.AxesImage at 0x1e115f0af50>
```



In [19]:

```
# Splitting the image in R,G,B arrays.

red,green,blue = cv2.split(img)
# it will split the original image into Blue, Green and Red arrays.

# initialize PCA with first 20 principal components
```

```
# change components to 100
pca = PCA(100)

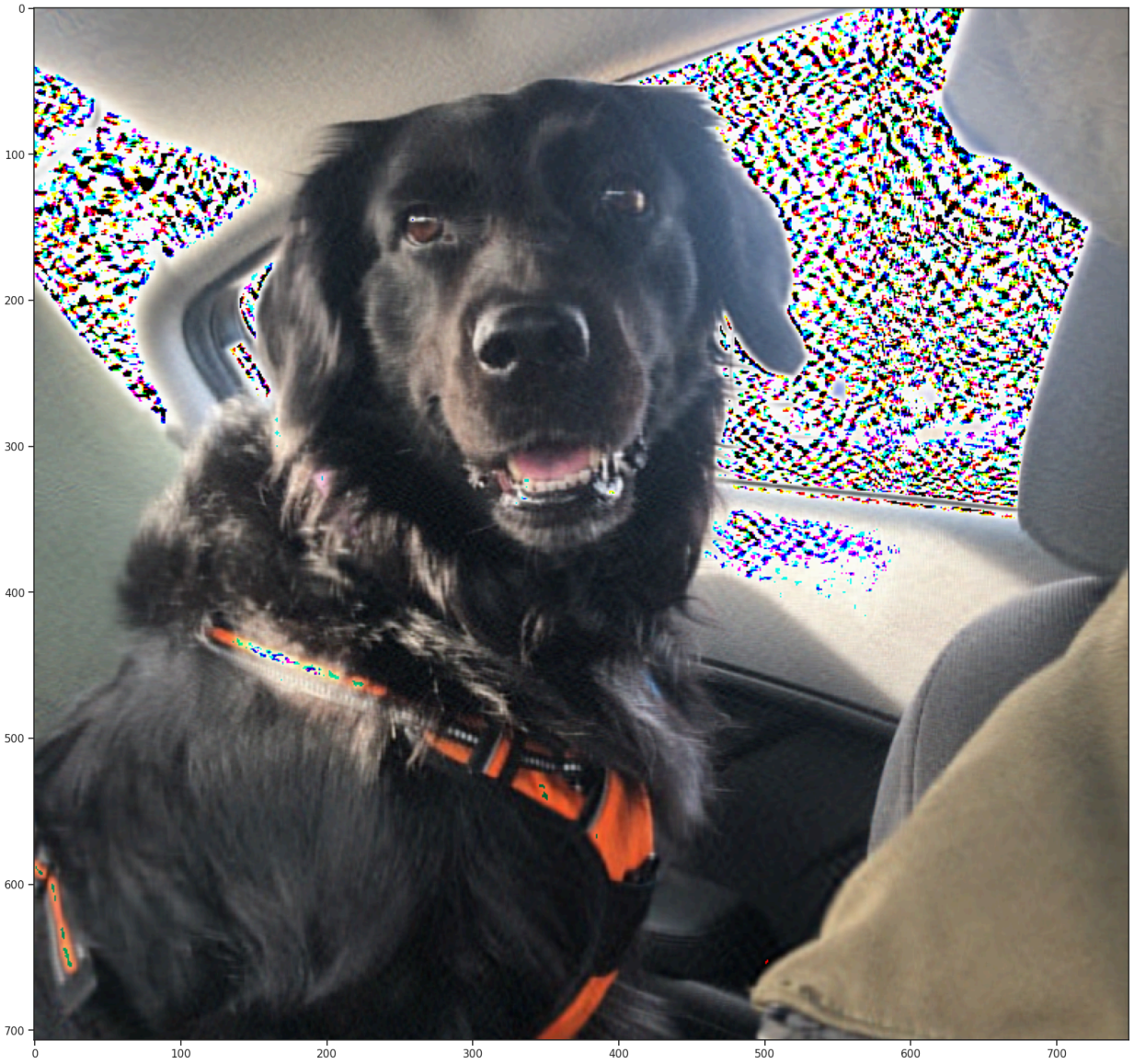
# Applying to red channel and then applying inverse transform to transformed array.
red_transformed = pca.fit_transform(red)
red_inverted = pca.inverse_transform(red_transformed)

#Applying to Green channel and then applying inverse transform to transformed array.
green_transformed = pca.fit_transform(green)
green_inverted = pca.inverse_transform(green_transformed)

#Applying to Blue channel and then applying inverse transform to transformed array.
blue_transformed = pca.fit_transform(blue)
blue_inverted = pca.inverse_transform(blue_transformed)

img_compressed = (np.dstack((red_inverted, green_inverted, blue_inverted))).astype(np.
#viewing the compressed image
plt.imshow(img_compressed)
```

Out[19]: <matplotlib.image.AxesImage at 0x1e113bac790>



In [ ]: