Visual Computing in the Life Sciences

Assignment sheet 6

Exercise 1 (Parallel Coordinates Plot in Plotly, 11 Points)

a) Use pandas to read the file Data_Cortex_Nuclear.xls available via eCampus. Extract subgroups t-CS-s and c-CS-s. How many mice were measured for each class? (1P)

In [1]:
```
import pandas as pd

path="Data_Cortex_Nuclear.xls"
data=pd.read_excel(path)
```

In [11]:
```
#data.info()
```

In [3]:
```
data.head()
```

Out[3]:

	MouseID	DYRK1A_N	ITSN1_N	BDNF_N	NR1_N	NR2A_N	pAKT_N	pBRAF_N	pCAMKII_
0	309_1	0.503644	0.747193	0.430175	2.816329	5.990152	0.218830	0.177565	2.37374
1	309_2	0.514617	0.689064	0.411770	2.789514	5.685038	0.211636	0.172817	2.29215
2	309_3	0.509183	0.730247	0.418309	2.687201	5.622059	0.209011	0.175722	2.28333
3	309_4	0.442107	0.617076	0.358626	2.466947	4.979503	0.222886	0.176463	2.15230
4	309_5	0.434940	0.617430	0.358802	2.365785	4.718679	0.213106	0.173627	2.13401

5 rows × 82 columns

In [4]:
```
df=pd.DataFrame(data)
df
```

Out[4]:
```
      MouseID  DYRK1A_N  ITSN1_N   BDNF_N     NR1_N    NR2A_N    pAKT_N   pBRAF_N  pCAM
   0    309_1  0.503644  0.747193  0.430175  2.816329  5.990152  0.218830  0.177565  2.37
   1    309_2  0.514617  0.689064  0.411770  2.789514  5.685038  0.211636  0.172817  2.29
   2    309_3  0.509183  0.730247  0.418309  2.687201  5.622059  0.209011  0.175722  2.28
   3    309_4  0.442107  0.617076  0.358626  2.466947  4.979503  0.222886  0.176463  2.15
   4    309_5  0.434940  0.617430  0.358802  2.365785  4.718679  0.213106  0.173627  2.13
 ...      ...       ...       ...       ...       ...       ...       ...       ...
1075 J3295_11  0.254860  0.463591  0.254860  2.092082  2.600035  0.211736  0.171262  2.48
1076 J3295_12  0.272198  0.474163  0.251638  2.161390  2.801492  0.251274  0.182496  2.51
1077 J3295_13  0.228700  0.395179  0.234118  1.733184  2.220852  0.220665  0.161435  1.98
1078 J3295_14  0.221242  0.412894  0.243974  1.876347  2.384088  0.208897  0.173623  2.08
1079 J3295_15  0.302626  0.461059  0.256564  2.092790  2.594348  0.251001  0.191811  2.36

1080 rows × 82 columns
```

In [5]:
```
df=data.loc[(df['class']=='c-CS-s') | (df['class']=='t-CS-s')]
df
```

Out[5]:

	MouseID	DYRK1A_N	ITSN1_N	BDNF_N	NR1_N	NR2A_N	pAKT_N	pBRAF_N	pCAM
300	3477_1	0.591618	0.814980	0.414980	2.706108	5.256264	0.260990	0.187249	4.1
301	3477_2	0.589807	0.840445	0.424325	2.803702	5.534373	0.269420	0.192834	4.2
302	3477_3	0.598902	0.886041	0.444760	2.918261	5.717712	0.290435	0.197162	4.4
303	3477_4	0.544844	0.726999	0.390846	2.521089	4.967323	0.275550	0.190962	4.5
304	3477_5	0.512785	0.730650	0.389176	2.514047	5.070519	0.271872	0.188625	4.5
...	...	...	...	...	...	...	...	...	
940	50810E_11	0.419345	0.643689	0.290638	2.266884	3.724422	0.196864	0.139730	2.6
941	50810E_12	0.439347	0.658009	0.296423	2.231571	3.676516	0.205599	0.141680	2.7
942	50810E_13	0.435826	0.674054	0.278855	2.071468	3.322622	0.208864	0.187073	2.2
943	50810E_14	0.424569	0.659863	0.276118	2.081833	3.193542	0.229356	0.164595	2.2
944	50810E_15	0.478822	0.666287	0.290978	2.147198	3.268946	0.224122	0.166192	2.3

240 rows × 82 columns

In [6]:
```
df['class'].value_counts()
```

Out[6]:
```
c-CS-s    135
t-CS-s    105
Name: class, dtype: int64
```

Class c-CS-s has 135 mice and class t-CS-s has 105.

b) Create a parallel coordinates plot with plotly from the following 5 proteins: (pPKCG N, pP70S6 N, pS6 N, pGSK3B N, ARC N). Assign different colors to the two selected classes. Annotate every axis with the correct protein name. (9P)

In [7]:
```
import plotly.express as px
```

In [8]:
```
new_df=df[['pPKCG_N', 'pP70S6_N','pS6_N', 'pGSK3B_N','ARC_N','class']]
new_df.reset_index()
```

Out[8]:

	index	pPKCG_N	pP70S6_N	pS6_N	pGSK3B_N	ARC_N	class
0	300	1.661613	0.331342	0.121155	0.169434	0.121155	c-CS-s
1	301	1.678407	0.361780	0.125333	0.169711	0.125333	c-CS-s
2	302	1.578564	0.388650	0.118600	0.178248	0.118600	c-CS-s
3	303	1.950057	0.445886	0.117714	0.161904	0.117714	c-CS-s
4	304	2.005071	0.447196	0.124239	0.163666	0.124239	c-CS-s
...	...	...	...	...	...	...	...
235	940	2.106951	0.344512	0.096220	0.163293	0.096220	t-CS-s
236	941	2.083990	0.342146	0.100739	0.158251	0.100739	t-CS-s
237	942	2.045874	0.295476	0.098713	0.165026	0.098713	t-CS-s
238	943	2.128058	0.312860	0.102199	0.166305	0.102199	t-CS-s
239	944	2.160519	0.327255	0.106035	0.176725	0.106035	t-CS-s

240 rows × 7 columns

In [9]:
```
f= lambda x: 1 if x=='c-CS-s' else 0  # setting class c-CS-s to 1 and t-CS-s to 0
new_df['class']=new_df['class'].map(f)
new_df
```

```
<ipython-input-9-2ac91cef82f9>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  new_df['class']=new_df['class'].map(f)
```

Out[9]:

	pPKCG_N	pP70S6_N	pS6_N	pGSK3B_N	ARC_N	class
300	1.661613	0.331342	0.121155	0.169434	0.121155	1
301	1.678407	0.361780	0.125333	0.169711	0.125333	1
302	1.578564	0.388650	0.118600	0.178248	0.118600	1
303	1.950057	0.445886	0.117714	0.161904	0.117714	1
304	2.005071	0.447196	0.124239	0.163666	0.124239	1
...	...	...	...	...	...	...
940	2.106951	0.344512	0.096220	0.163293	0.096220	0
941	2.083990	0.342146	0.100739	0.158251	0.100739	0
942	2.045874	0.295476	0.098713	0.165026	0.098713	0
943	2.128058	0.312860	0.102199	0.166305	0.102199	0
944	2.160519	0.327255	0.106035	0.176725	0.106035	0

240 rows × 6 columns

```
In [12]: fig = px.parallel_coordinates(new_df, color="class",
                                       dimensions=['pPKCG_N', 'pP70S6_N','pS6_N', 'pGSK3B_N','AR
         C_N'],
                                      color_continuous_scale=px.colors.diverging.Tealrose,
                                      color_continuous_midpoint=0.5)
         fig.show()
```

c) Explore the data by interacting with the parallel coordinates plot. Do you find anything suspicious about the data set? (1P)

I don't know, there may be some repeating data in the data set?

Exercise 2 (Comparing RadViz and Star Coordinates, 14 Points)

a) As we mentioned in the lecture, the star coordinates, and RadViz are two popular multidimensional data visualization techniques. Why can they also be interpreted as dimensionality reduction techniques? (1P)

In order to present high dimensional data as graphs, many multidimensional data visualization techniques (like the star coordinates and RadViz) define procedures that can map numerical multivariate data onto low dimention form. So they can also be interpreted as dimensionality reduction techniques

b) In general, the authors of this paper prefer star coordinates over RadViz. Briefly explain two reasons for their preference. (2P)

- The normalization step in RadViz introduces nonlinear distotions when dealing with data. These nonlinear distortions can interrupt outlier detections, make it difficult to associate the plots with other useful linear mappings and affect the accuracy of estimating original data attributes.
- Also, there is less flexibility when choosing different layouts and views in RadViz.

c) Despite this, the authors mention a specific use case where they consider RadViz to be superior to star coordinates. Briefly explain what this use case is and why they prefer RadViz in this case. (2P)

The use case is when the data samples contains only a few related attributes which are larger than the other attributes. Because of the extra normalization step in RadViz, it is superior to star coordinates when using in this kind of cases.

d) Neither RadViz nor star coordinates provide a one-to-one mapping between a data point's high-dimensional location and its two-dimensional projection. Interactively modifying the anchor points or axes, respectively, can reduce the resulting ambiguities. However, the paper mentions two examples in which, unlike star coordinates, RadViz introduces ambiguities that cannot be resolved with any rearrangement of the anchor points. Point out these two examples. Could the extended RadViz that we learned about in the lecture resolve them? (3P)

- RadViz introduces ambiguities in the estimation of original data attributes and arbitrary layouts.
- The extended RadViz can't solve these problems. Because those ambiguities are caused by the normalization step in RadViz, the extended RadViz doesn't solve the problems brought by the extr normalization.

e) Several methods have been proposed to optimize the locations of anchor points in RadViz to obtain an improved separation of classes in the resulting projection. How did two such algorithms compare to supervised linear dimensionality reduction techniques in the experiments reported in this paper, in terms of (i) computational effort and (ii) achieved class separation? (2P)

The computational effort is much larger when using these two algorithms and regular configurations are often not enough to get a satisfied class seperation. On the contrary, other dimensionality reduction methods that can generate linear mappings can seperate classes optimally.

f) The authors propose that star coordinates could be combined with Linear Discriminant Analysis in order to perform a manual feature selection. Briefly explain how that approach works. (2P)

To correctly select features is time-consuming and complex,it is also not enough to only use regular configurations of star coordinates. Combining other linear dimensionality reduction methods can generate linear mappings. Matrices related to the mappings can be computed automatically and thus specifies the axis vectors that can be used.

g) Briefly explain why anchor points in RadViz are commonly chosen such that they form a corner of the overall convex hull. Is this property also met for the star coordinate axis configuration in Fig. 12 (a)? Is it true in Fig. 12 (b)? (2P)

The anchor points are chosen in this way in order to analyze sparse data effectively. This property isn't met for the star coordinate axis configuration in Fig. 12 (a) but I think it is true in Fig. 12 (b).

Exercise 3 (Principal Component Analysis, 17 Points)

In [12]:
```
import pandas as pd
import numpy as np
import scipy
import scipy.interpolate
import scipy.stats
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
import seaborn as sns
from sklearn.decomposition import PCA
from sklearn.manifold import Isomap
from sklearn.manifold import TSNE
from pandas.core.frame import DataFrame
from pandas.core.series import Series
pd.options.mode.chained_assignment = None
```

a) Write a program to read the breast-cancer-wisconsin.xlsx file again. Interpolate missing values as before, but keep all variables this time. Perform a Principal Component Analysis (PCA) on the values. (2P) Make a plot that, for any number n, shows what fraction of the overall variance in the data is contained in the first n principal components. (2P) How many components do we need to cover >= 90% of the variance? (1P)

In [13]:
```
data = pd.read_excel('breast-cancer-wisconsin.xlsx')
data.head()
```

Out[13]:

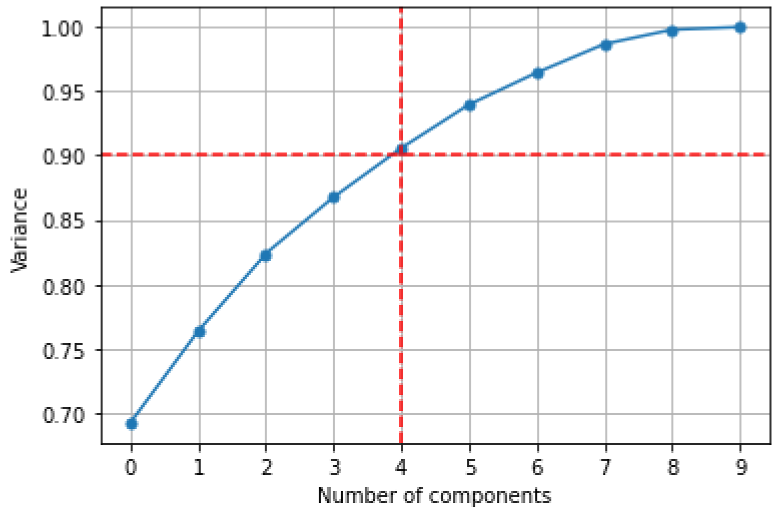
	code	thickness	uniCelS	uniCelShape	marAdh	epiCelSize	bareNuc	blaChroma	normNu
0	1000025	5	1	1	1	2	1.0	3	
1	1002945	5	4	4	5	7	10.0	3	
2	1015425	3	1	1	1	2	2.0	3	
3	1016277	6	8	8	1	3	4.0	3	
4	1017023	4	1	1	3	2	1.0	3	

In [14]:
```
def interpolate_column(data: DataFrame, column: str) -> Series:
    samples_with_missing_values_list = data[data.loc[:, column].isna()].index.tolist()
    data_subset = data.drop(samples_with_missing_values_list)
    predictor = scipy.interpolate.NearestNDInterpolator(data_subset.drop(column,axis=1
), data_subset[column])
    predicted_column = predictor(data.drop(columns=column, axis=1))
    result_column = data[column]
    result_column[samples_with_missing_values_list] =predicted_column[samples_with_miss
ing_values_list]
    return result_column
def show_pca_variance_plot(explained_variance_ratio: np.ndarray) -> None:
    components_nums = range(len(explained_variance_ratio))
    fig, ax = plt.subplots(1, 1)
    ax.plot(components_nums, explained_variance_ratio, marker=".", markersize=10)
    ax.xaxis.set_major_locator(ticker.MultipleLocator(1))
    ax.set_xlabel("Number of components")
    ax.set_ylabel("Variance")
    ax.axvline(4, linestyle='--', color='red')
    ax.axhline(.9, linestyle='--', color='red')
    ax.grid()
```

In [15]:
```
data_cut = data.drop(columns = "code")
data["bareNuc"] = interpolate_column(data_cut, "bareNuc")
data["bareNuc"].isna().sum()
```

Out[15]:
```
0
```

In [16]:
```
pca = PCA()
pca.fit(data.drop(columns="code"))
explained_variance_ratio = pca.explained_variance_ratio_.cumsum()
show_pca_variance_plot(explained_variance_ratio)
```



We need at least 5 components to cover ≥ 90% of the variance.

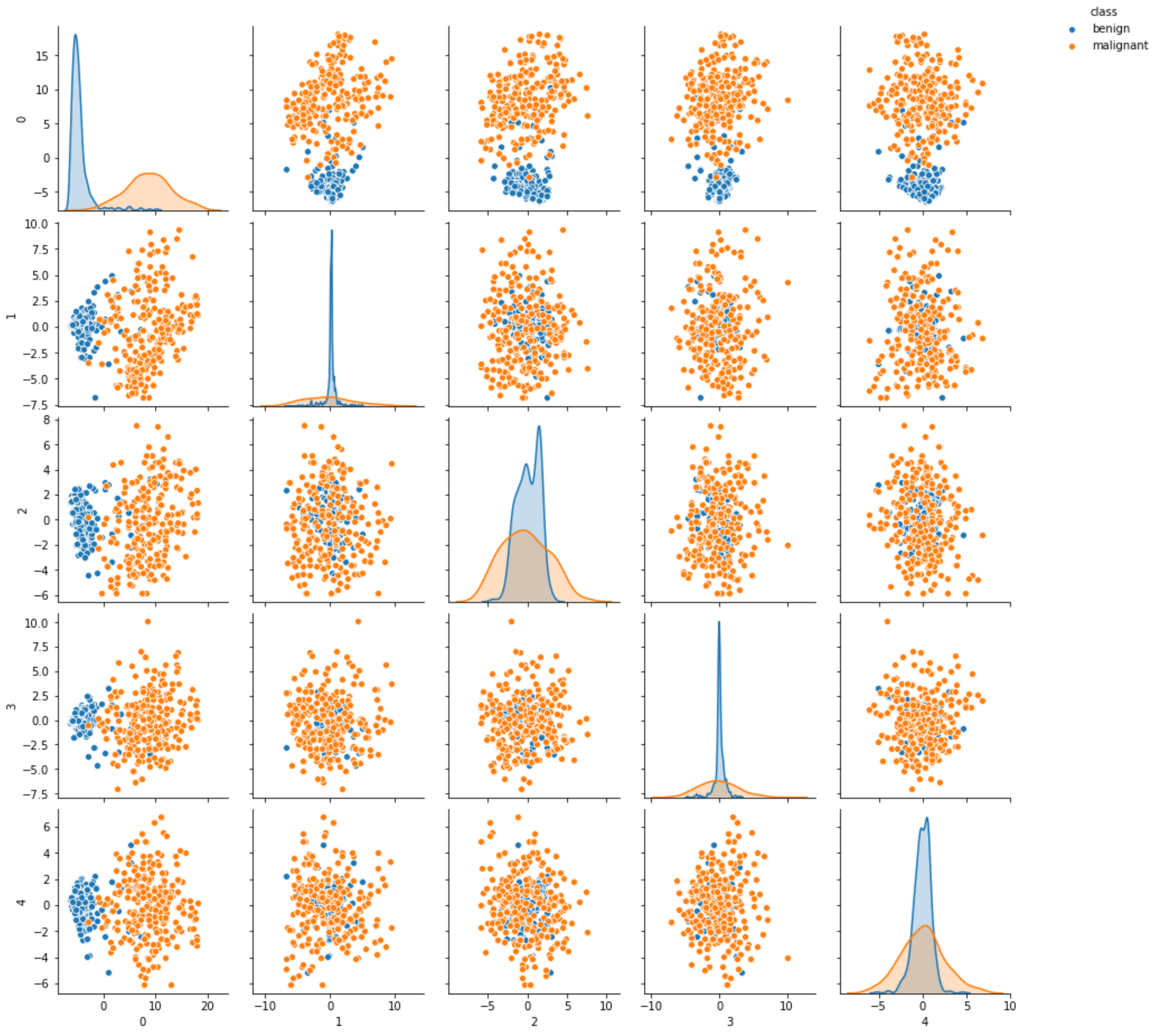
b) Each sample is now characterized by a point in PCA space. Create a scatter plot matrix that shows the first five principal components. Each diagonal cell should contain two overlaid density plots, one for the benign and one for the malignant class. Use different colors to distinguish between the classes, and add a legend that clearly states which samples are benign or malignant. (3P)

In [17]:
```
def run_pca(data: DataFrame, n_components: int, class_attr: str = "class")->DataFrame:
    data_cut = data.drop(columns=class_attr)
    pca = PCA(n_components=n_components)
    transformed_data = pca.fit(data_cut).transform(data_cut)
    transformed_data = pd.DataFrame(data=transformed_data)
    transformed_data[class_attr] = data[class_attr].reset_index()[class_attr]
    loadings = pca.components_
    header = ["PC{0}".format(x + 1) for x in range(loadings.shape[0])]
    df_loadings = pd.DataFrame(data=loadings, index=header, columns=data_cut.columns)
    return (transformed_data, df_loadings)
def show_pairplot(data: DataFrame, class_labels=None) -> None:
    g = sns.pairplot(data, hue="class")

    if class_labels:
        for t in g._legend.texts:
            t.set_text(class_labels[t.get_text()])
    g._legend.set_bbox_to_anchor((1.1, 1))
```

In [18]:
```
transformed_data = run_pca(data_cut, 5)[0]
# transformed_data.head()
```

In [19]:
```
CLASS_LABELS = {
"2": "benign",
"4": "malignant"
}
show_pairplot(transformed_data, CLASS_LABELS)
```



c) Which PCA mode shows the strongest difference between the benign and the malignant samples? Name the original variables that have the highest and lowest weights in its definition, respectively. (3P)

The PCA mode “0” shows the strongest difference between the benign and the malignant samples. The original variables that have the highest and lowest wights: “bareNuc” and “mitoses” respectively.

In [20]:
```
run_pca(data_cut, 5)[1].head(1)
```

Out[20]:

	thickness	uniCelS	uniCelShape	marAdh	epiCelSize	bareNuc	blaChroma	normNuc	m
PC1	0.297623	0.40339	0.391298	0.330883	0.249457	0.443425	0.29176	0.356387	0.

d) Scatterplot matrices often reveal outliers in the data. Visually identify at least one sample that is far away from the others, remove it from the dataset, and re-generate the scatterplot matrix without it. (3P)

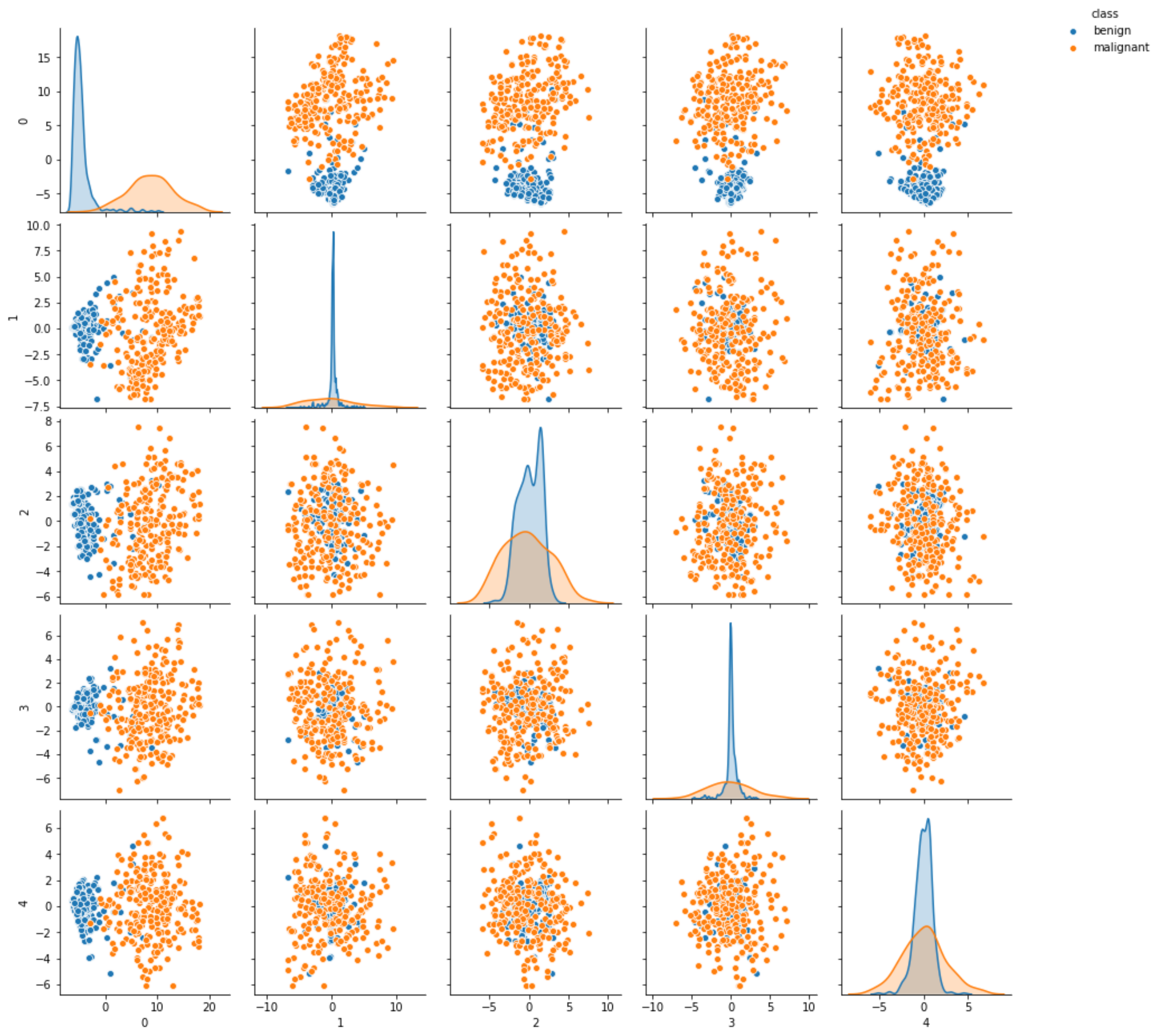
Visually, there is 1 outlier for the component “3” which have unusual values exceeding 8.

In [21]:
```
outlier = transformed_data.loc[transformed_data[3] > 8, :]
outlier
```

Out[21]:

	0	1	2	3	4	class
167	8.478326	4.32948	-2.026009	10.099392	-4.017767	4

In [22]:
```
transformed_data_without_outlier = transformed_data.drop(outlier.index)
show_pairplot(transformed_data_without_outlier, CLASS_LABELS)
```



e) In the breast cancer dataset, all variables xi have a similar range, xi ∈ [1; 10]. If the variables of a dataset have very different ranges, for example one variable x1 ∈ [1000; 2000] and another one x2 ∈ [1; 5], how would this affect the PCA? Could it make sense to pre-process the data in such cases? Why and how? (3P)

If the variables of a dataset have very different ranges, then one of these variables will not affect the principal component value, and the principal component will reflect only the value of the other variable. In this case it can be very useful to normalize values onto the same values range (e.g. [0; 1]).

Exercise 4 (Pitfalls in t-SNE, 8 Points)

a) Pick the "three clusters with equal numbers of points" data set. Set the number of points per class to 10, and number of dimensions to 50. Run the demo once with perplexity=29, and once with perplexity=30. Explain why there is a big difference in the final 2D embedding? (2P)

Perplexity is greater than number of points per class. Slightly changing of the perplexity may have huge effect on the reault as it is a global parameter. Still, the points are too few while the dimension is large, this may also be a reason for the big difference.

b) Try the example "a square grid with equal spacing between points", with 20 points per side. In the resulting plot with perplexity=100, why are distances between points in the middle of the square larger than near the boundary? (3P)

Distance of well-seperated points may mean nothing in a t-SNE plot.T-SNE tends to expand the denser regions of a group of data. Although we know the points are evenly distributed, t-SNE recognizes the middles of the clusters have less empty space around them than the ends. So the it magnifies those in the middle.

c) Pick "a square grid with equal spacing between points" data set, with 20 points per side, and perplexity=2. Run t-SNE multiple times. You will observe that the square grid sometimes breaks down into separate smaller clusters. Why? (3P)

We can sometimes see patterns in what is really just random data, those "clumps" are not meaningful. particularly, low perplexity values often lead to this kind of distribution. When at low perplexity, local effects and meaningless “clumping” take center stage of the plot, resulting misleadings.