Machine Learning - Laboratory 3

```
import getopt
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import axes3d
from matplotlib import cm
from pdffuns import *
import pickle

def labsol3(met='ML', discr='pxw', prm = []):

    # Initialise values:
    # - axes, x1 and x2
    x1 = np.arange(-6.25,6.26,0.5).reshape(-1,1)
    x2 = np.arange(-6.25,6.26,0.5).reshape(-1,1)

    # Get coordinates grid
    X1, X2 = np.meshgrid(x1, x2)

    # Pack everything
    Xgrid = np.dstack((X1, X2))

    # Load data from pickle files
    pfile = 'lab3.p'
    with open(pfile, "rb") as fp:
        X=pickle.load(fp)

    # Number of classes
    M = len(X)

    # Feature dimension
    l = len(X[0])

    # Estimate prior probabilities, Pwi[k].
    N, Pw = [], []

    # Number of feature vectors: N[i] --> Feature vectors of class i
    for i in range(M):
        N.append(len(X[i][0]))

    # Determine Pwi
    for i in range(M):
        Pw.append(N[i]/sum(N))


    # Initialise method specific parameters
```

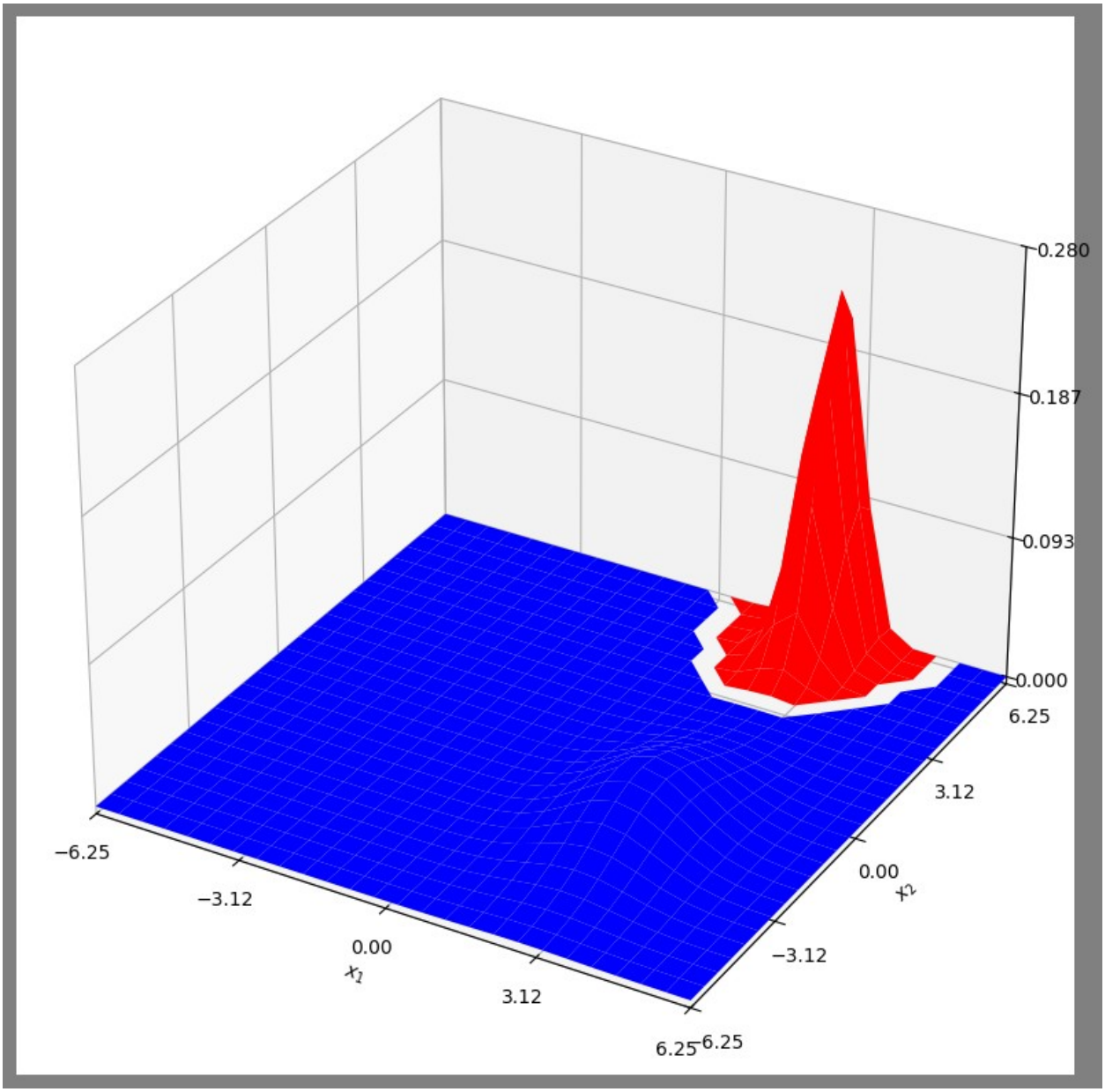
```
    # - on condition of met
    # - Maximum likelihood: my, Sgm (empty)
    # - Parzen window: h1 from prm
    # - kn nearest neighbor: knn from prm
    if met == 'ML':
        my = np.zeros(shape=(M, l), dtype=float)
        Sgm = np.zeros(shape=(M, l, l), dtype=float)
    if met == 'knn':
        my = None
        Sgm = None
        kn = prm[0]
    if met == 'PZ':
        my = None
        Sgm = None
        h1 = prm[0]
    # - parameters, my[i] and Sgm[i], i = 0,...,M-1
    # - prior probabilities, Pw[i], i = 0,...,M-1

    # Determine class specific probability density functions, pxw[i],
i = 0,...,M-1
    # - initialise pxw as empty list
    pxw = np.zeros(shape=(M, np.shape(Xgrid)[0], np.shape(Xgrid)[1]))

    g = np.zeros(shape=pxw.shape)
    # - initialise total density function, p as zero
    p = 0
    # - iterate over classes, k = 0,...,M-1
    for i in range(M):

        # - Maximum likelihood:
        if met == 'ML':
            # - estimate parameters my[k], Sgm[k]
            for j in range(l):
                my[i][j] = np.mean(X[i][j])

            Sgm[i] = np.cov(X[i])

            # - determine pxw[k] by using norm2D
            # Reshape 'my' to pass it as a column vector
            pxw[i, :, :] = norm2D(my[i].reshape(-1, 1), Sgm[i], Xgrid)

        # - kn-nearest neighbor:
        if met == "knn":
            # - use knn2D to determine pxw[k] from X[k]
            pxw[i, :, :] = knn2D(X[i], Xgrid, kn)

        # - Parzen window:
        if met == "PZ":
            hn = h1/np.sqrt(N[i])
            hnI = hn**2 * np.eye(l)
```

```
            # iterate over all feature vectors in class i
            for j in range(0, N[i]):
                # feature vector j of class i
                xk = X[i][:,j].reshape(l,1)
                # sum up the probabilities of each of the N[i]
distributions
                # Note that there is one distribution for each
datapoint!
                pxw[i, :, :] = pxw[i, :, :] + norm2D(xk, hnI, Xgrid)
                # divide by number of feature vectors in class i
                pxw[i, :, :] /= N[i]

        # - update p
        p += Pw[i] * pxw[i]

    # Determine discriminant functions, g[k], k = 0,...,M-1
    for i in range(M):
        g[i] = (Pw[i] * pxw[i])

    return x1, x2, my, Sgm, g/p, g
```

Sections a), b), c) and d)

```
# Call the function to get the data
x1, x2, my, Sgm, posterior, df= labsol3()

# Print estimated parameters
print(f"Estimated mean vectors: \n{my[0].reshape(-1, 1)} \n
{my[1].reshape(-1, 1)}")
print(f"Estimated covariance matrices: \n{Sgm}")

# Plot the discriminant functions
classplot(df, x1, x2, 1, gsv={'gsv': 1, 'figstr': 'pdf'})
```

```
Estimated mean vectors:
[[2.65 ]
 [5.825]]
 [[ 2.8       ]
 [-1.76666667]]
Estimated covariance matrices:
[[[ 0.19666667  0.005     ]
  [ 0.005       0.73583333]]

 [[ 1.11       -0.01      ]
  [-0.01        2.92333333]]]
```

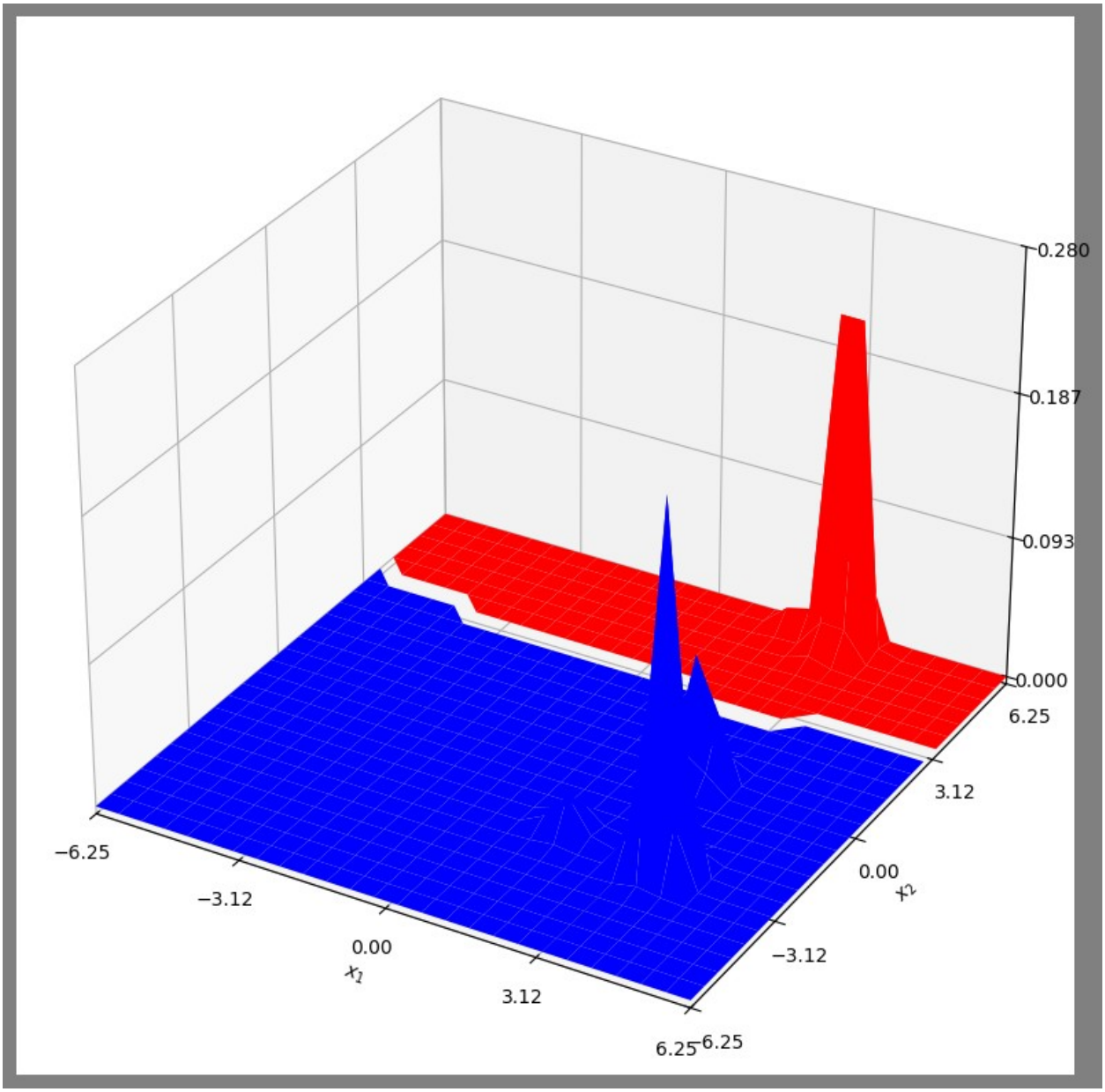


Section e)

If we compare this discriminant functions to the ones that we got in **labsol2.ipynb**, we can notice they are quite similar. Nevertheless, we can also see that the decision border is less smoothed, this could be directly caused by the few amount of data we have to estimate the parameters μ and Σ.

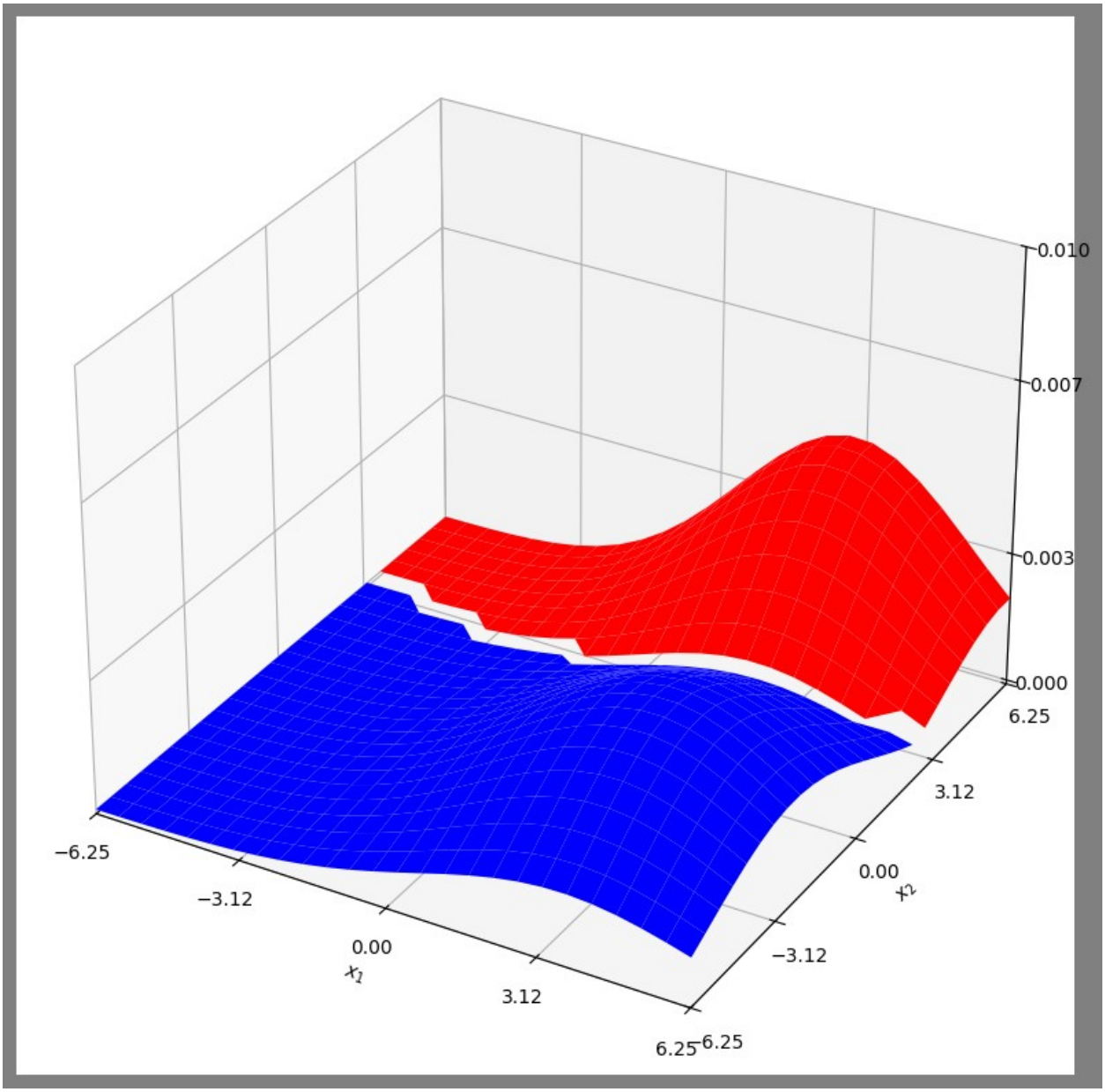
Section f)

```
x1, x2, my, Sgm, posterior, df = labsol3('PZ', 'pxw', [0.5])

classplot(df, x1, x2, 1, gsv={'gsv': 1, 'figstr': 'pdf'})
```



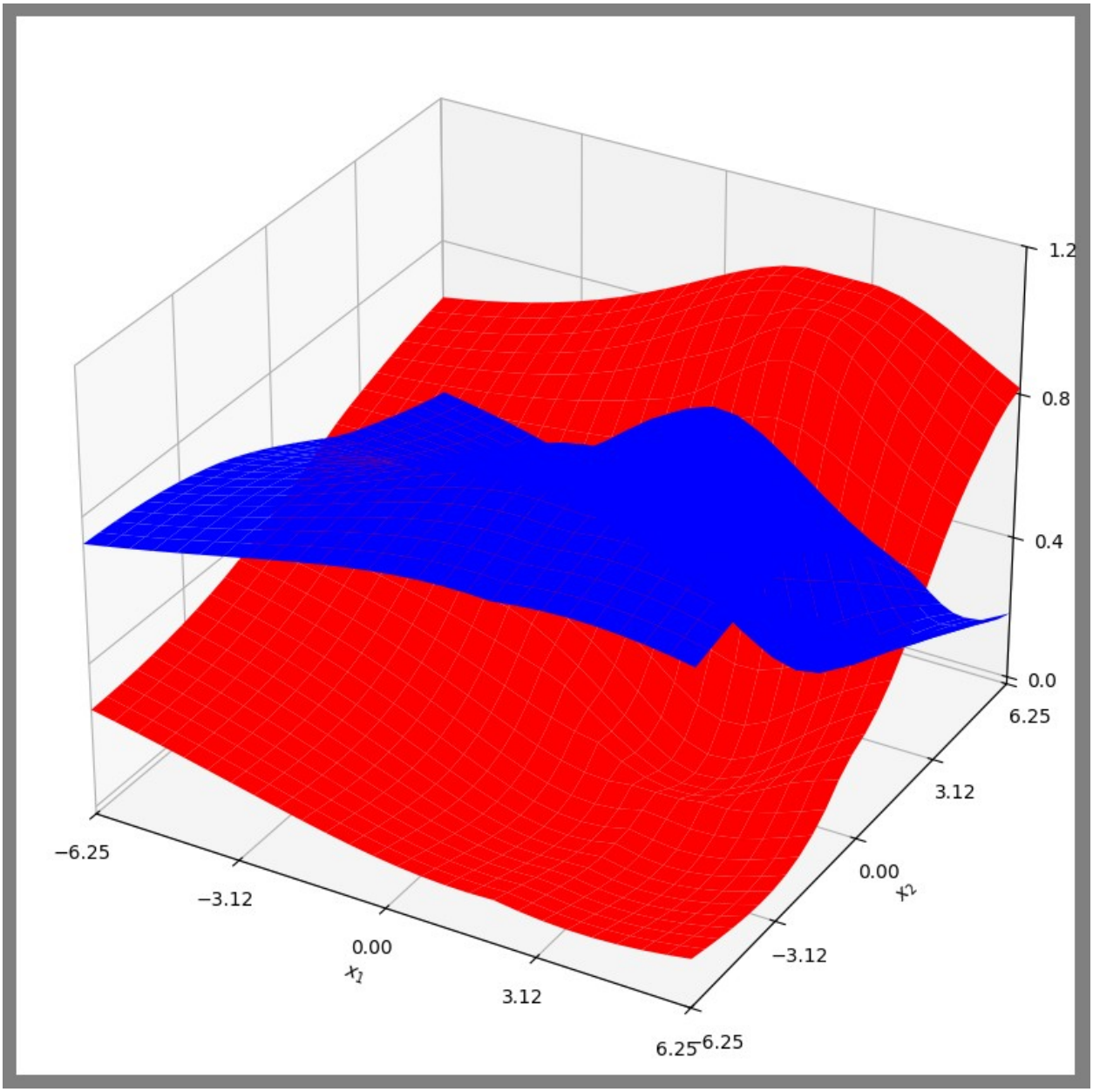
Section g)

```
# Call the function to get the data
x1, x2, my, Sgm, posterior, df = labsol3('PZ', 'pxw', [5.])

classplot(df, x1, x2, 1, gsv={'gsv': 1, 'figstr': 'pdf'})
```



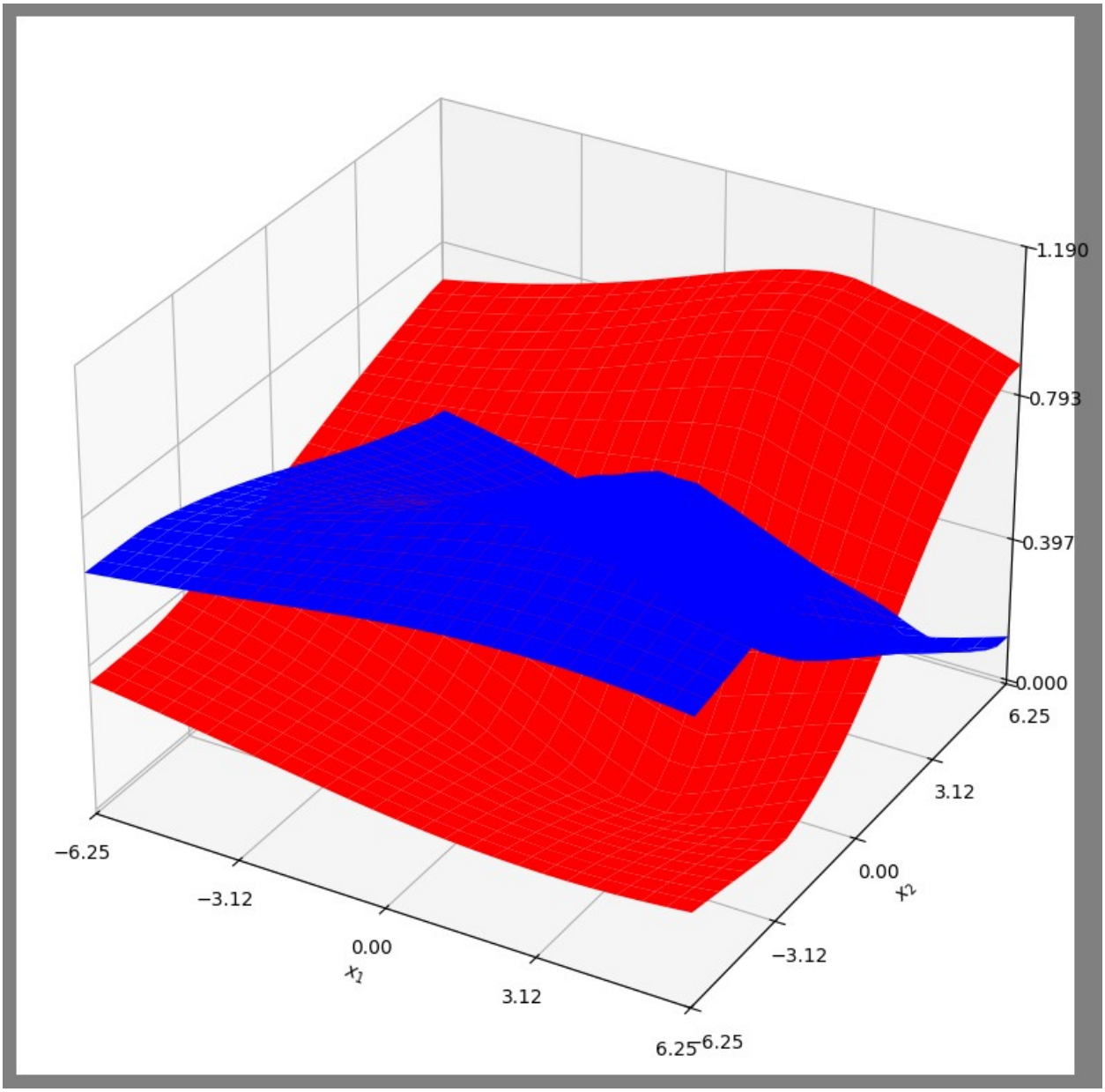
Section h)

```
# Call the function to get the data
x1, x2, my, Sgm, posterior, df= labsol3('knn', 'pxw', [1])

# Plot the discriminant functions
classplot(posterior, x1, x2, 0, gsv={'gsv': 1, 'figstr': 'pdf'})
```



Section i)

```
# Call the function to get the data
x1, x2, my, Sgm, posterior, df = labsol3('knn', 'pxw', [3])

# Plot the discriminant functions
classplot(posterior, x1, x2, 0, gsv={'gsv': 1, 'figstr': 'pdf'})
```



Section j)

```
# Call the function to get the data
x1, x2, my, Sgm, posterior, df = labsol3('knn', 'pxw', [5])

# Plot the discriminant functions
classplot(posterior, x1, x2, 0, gsv={'gsv': 1, 'figstr': 'pdf'})
```

```
Kn can't be bigger than the size of the dataset
Kn can't be bigger than the size of the dataset
```

```
C:\Users\danie\AppData\Local\Temp\ipykernel_17332\656256274.py:105:
RuntimeWarning: invalid value encountered in divide
  return x1, x2, my, Sgm, g/p, g

---------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
Cell In[18], line 5
      2 x1, x2, my, Sgm, posterior, df = labsol3('knn', 'pxw', [5])
      4 # Plot the discriminant functions
----> 5 classplot(posterior, x1, x2, 0, gsv={'gsv': 1, 'figstr':
'pdf'})

File c:\Users\danie\Desktop\Ing. Informatica\CUARTO\ELE520_ML\
Assignments\Lab\lab3\pdffuns.py:124, in classplot(g, x1, x2, gnan,
discr, gsv)
    122 ax.set(xlabel='$x_1$', ylabel='$x_2$', zlabel=zlb)
    123 if gsv['gsv']:
--> 124     plt.savefig(gsv['figstr'] + discr + '.png')
    126 plt.show()
    128 return

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
pyplot.py:1228, in savefig(*args, **kwargs)
   1225 fig = gcf()
   1226 # savefig default implementation has no return, so mypy is
unhappy
   1227 # presumably this is here because subclasses can return?
-> 1228 res = fig.savefig(*args, **kwargs)  # type: ignore[func-
returns-value]
   1229 fig.canvas.draw_idle()  # Need this if 'transparent=True', to
reset colors.
   1230 return res

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
figure.py:3395, in Figure.savefig(self, fname, transparent, **kwargs)
   3393     for ax in self.axes:
   3394         _recursively_make_axes_transparent(stack, ax)
-> 3395 self.canvas.print_figure(fname, **kwargs)

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
backend_bases.py:2204, in FigureCanvasBase.print_figure(self,
filename, dpi, facecolor, edgecolor, orientation, format, bbox_inches,
pad_inches, bbox_extra_artists, backend, **kwargs)
   2200 try:
   2201     # _get_renderer may change the figure dpi (as vector
formats
   2202     # force the figure dpi to 72), so we need to set it again
here.
```

```
   2203     with cbook._setattr_cm(self.figure, dpi=dpi):
-> 2204         result = print_method(
   2205             filename,
   2206             facecolor=facecolor,
   2207             edgecolor=edgecolor,
   2208             orientation=orientation,
   2209             bbox_inches_restore=_bbox_inches_restore,
   2210             **kwargs)
   2211 finally:
   2212     if bbox_inches and restore_bbox:

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
backend_bases.py:2054, in
FigureCanvasBase._switch_canvas_and_return_print_method.<locals>.<lamb
da>(*args, **kwargs)
   2050     optional_kws = {  # Passed by print_figure for other
renderers.
   2051         "dpi", "facecolor", "edgecolor", "orientation",
   2052         "bbox_inches_restore"}
   2053     skip = optional_kws -
{*inspect.signature(meth).parameters}
-> 2054     print_method = functools.wraps(meth)(lambda *args,
**kwargs: meth(
   2055         *args, **{k: v for k, v in kwargs.items() if k not in
skip}))
   2056 else:  # Let third-parties do as they see fit.
   2057     print_method = meth

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
backends\backend_agg.py:496, in FigureCanvasAgg.print_png(self,
filename_or_obj, metadata, pil_kwargs)
    449 def print_png(self, filename_or_obj, *, metadata=None,
pil_kwargs=None):
    450     """
    451     Write the figure to a PNG file.
    452
   (...)
    494         *metadata*, including the default 'Software' key.
    495     """
--> 496     self._print_pil(filename_or_obj, "png", pil_kwargs,
metadata)

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
backends\backend_agg.py:444, in FigureCanvasAgg._print_pil(self,
filename_or_obj, fmt, pil_kwargs, metadata)
    439 def _print_pil(self, filename_or_obj, fmt, pil_kwargs,
metadata=None):
    440     """
    441     Draw the canvas, then save it using `.image.imsave` (to
which
```

```
    442     *pil_kwargs* and *metadata* are forwarded).
    443     """
--> 444     FigureCanvasAgg.draw(self)
    445     mpl.image.imsave(
    446         filename_or_obj, self.buffer_rgba(), format=fmt,
origin="upper",
    447         dpi=self.figure.dpi, metadata=metadata,
pil_kwargs=pil_kwargs)

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
backends\backend_agg.py:387, in FigureCanvasAgg.draw(self)
    384 # Acquire a lock on the shared font cache.
    385 with (self.toolbar._wait_cursor_for_draw_cm() if self.toolbar
    386       else nullcontext()):
--> 387     self.figure.draw(self.renderer)
    388     # A GUI class may be need to update a window using this
draw, so
    389     # don't forget to call the superclass.
    390     super().draw()

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
artist.py:95, in _finalize_rasterization.<locals>.draw_wrapper(artist,
renderer, *args, **kwargs)
     93 @wraps(draw)
     94 def draw_wrapper(artist, renderer, *args, **kwargs):
---> 95     result = draw(artist, renderer, *args, **kwargs)
     96     if renderer._rasterizing:
     97         renderer.stop_rasterizing()

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
artist.py:72, in allow_rasterization.<locals>.draw_wrapper(artist,
renderer)
     69     if artist.get_agg_filter() is not None:
     70         renderer.start_filter()
---> 72     return draw(artist, renderer)
     73 finally:
     74     if artist.get_agg_filter() is not None:

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
figure.py:3162, in Figure.draw(self, renderer)
   3159             # ValueError can occur when resizing a window.
   3161     self.patch.draw(renderer)
-> 3162     mimage._draw_list_compositing_images(
   3163         renderer, self, artists, self.suppressComposite)
   3165     renderer.close_group('figure')
   3166 finally:

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
image.py:132, in _draw_list_compositing_images(renderer, parent,
artists, suppress_composite)
```

```
    130 if not_composite or not has_images:
    131     for a in artists:
--> 132         a.draw(renderer)
    133 else:
    134     # Composite any adjacent images together
    135     image_group = []

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
artist.py:72, in allow_rasterization.<locals>.draw_wrapper(artist,
renderer)
     69     if artist.get_agg_filter() is not None:
     70         renderer.start_filter()
---> 72     return draw(artist, renderer)
     73 finally:
     74     if artist.get_agg_filter() is not None:

File ~\AppData\Roaming\Python\Python312\site-packages\mpl_toolkits\
mplot3d\axes3d.py:460, in Axes3D.draw(self, renderer)
    458 # Then gridlines
    459 for axis in self._axis_map.values():
--> 460     axis.draw_grid(renderer)
    461 # Then axes, labels, text, and ticks
    462 for axis in self._axis_map.values():

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
artist.py:72, in allow_rasterization.<locals>.draw_wrapper(artist,
renderer)
     69     if artist.get_agg_filter() is not None:
     70         renderer.start_filter()
---> 72     return draw(artist, renderer)
     73 finally:
     74     if artist.get_agg_filter() is not None:

File ~\AppData\Roaming\Python\Python312\site-packages\mpl_toolkits\
mplot3d\axis3d.py:649, in Axis.draw_grid(self, renderer)
    645     return
    647 renderer.open_group("grid3d", gid=self.get_gid())
--> 649 ticks = self._update_ticks()
    650 if len(ticks):
    651     # Get general axis information:
    652     info = self._axinfo

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
axis.py:1301, in Axis._update_ticks(self)
   1296 """
   1297 Update ticks (position and labels) using the current data
interval of
   1298 the axes.  Return the list of ticks that will be drawn.
   1299 """
   1300 major_locs = self.get_majorticklocs()
```

```
-> 1301 major_labels = self.major.formatter.format_ticks(major_locs)
   1302 major_ticks = self.get_major_ticks(len(major_locs))
   1303 for tick, loc, label in zip(major_ticks, major_locs,
major_labels):

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
ticker.py:216, in Formatter.format_ticks(self, values)
    214 def format_ticks(self, values):
    215     """Return the tick labels for all the ticks at once."""
--> 216     self.set_locs(values)
    217     return [self(value, i) for i, value in enumerate(values)]

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
ticker.py:728, in ScalarFormatter.set_locs(self, locs)
    726     self._compute_offset()
    727 self._set_order_of_magnitude()
--> 728 self._set_format()

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
ticker.py:821, in ScalarFormatter._set_format(self)
    818 if len(self.locs) < 2:
    819     # We needed the end points only for the loc_range
calculation.
    820     locs = locs[:-2]
--> 821 loc_range_oom = int(math.floor(math.log10(loc_range)))
    822 # first estimate:
    823 sigfigs = max(0, 3 - loc_range_oom)

ValueError: cannot convert float NaN to integer

Error in callback <function _draw_all_if_interactive at
0x0000026DF0B9CAE0> (for post_execute), with arguments args (),kwargs
{}:

---------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
pyplot.py:268, in _draw_all_if_interactive()
    266 def _draw_all_if_interactive() -> None:
    267     if matplotlib.is_interactive():
--> 268         draw_all()

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
_pylab_helpers.py:131, in Gcf.draw_all(cls, force)
    129 for manager in cls.get_all_fig_managers():
    130     if force or manager.canvas.figure.stale:
--> 131         manager.canvas.draw_idle()
```

```
File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
backend_bases.py:1905, in FigureCanvasBase.draw_idle(self, *args,
**kwargs)
   1903 if not self._is_idle_drawing:
   1904     with self._idle_draw_cntx():
-> 1905         self.draw(*args, **kwargs)

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
backends\backend_agg.py:387, in FigureCanvasAgg.draw(self)
    384 # Acquire a lock on the shared font cache.
    385 with (self.toolbar._wait_cursor_for_draw_cm() if self.toolbar
    386       else nullcontext()):
--> 387     self.figure.draw(self.renderer)
    388     # A GUI class may be need to update a window using this
draw, so
    389     # don't forget to call the superclass.
    390     super().draw()

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
artist.py:95, in _finalize_rasterization.<locals>.draw_wrapper(artist,
renderer, *args, **kwargs)
     93 @wraps(draw)
     94 def draw_wrapper(artist, renderer, *args, **kwargs):
---> 95     result = draw(artist, renderer, *args, **kwargs)
     96     if renderer._rasterizing:
     97         renderer.stop_rasterizing()

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
artist.py:72, in allow_rasterization.<locals>.draw_wrapper(artist,
renderer)
     69     if artist.get_agg_filter() is not None:
     70         renderer.start_filter()
---> 72     return draw(artist, renderer)
     73 finally:
     74     if artist.get_agg_filter() is not None:

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
figure.py:3162, in Figure.draw(self, renderer)
   3159             # ValueError can occur when resizing a window.
   3161     self.patch.draw(renderer)
-> 3162     mimage._draw_list_compositing_images(
   3163         renderer, self, artists, self.suppressComposite)
   3165     renderer.close_group('figure')
   3166 finally:

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
image.py:132, in _draw_list_compositing_images(renderer, parent,
artists, suppress_composite)
    130 if not_composite or not has_images:
    131     for a in artists:
```

```
--> 132         a.draw(renderer)
    133 else:
    134     # Composite any adjacent images together
    135     image_group = []

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
artist.py:72, in allow_rasterization.<locals>.draw_wrapper(artist,
renderer)
     69     if artist.get_agg_filter() is not None:
     70         renderer.start_filter()
---> 72     return draw(artist, renderer)
     73 finally:
     74     if artist.get_agg_filter() is not None:

File ~\AppData\Roaming\Python\Python312\site-packages\mpl_toolkits\
mplot3d\axes3d.py:460, in Axes3D.draw(self, renderer)
    458 # Then gridlines
    459 for axis in self._axis_map.values():
--> 460     axis.draw_grid(renderer)
    461 # Then axes, labels, text, and ticks
    462 for axis in self._axis_map.values():

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
artist.py:72, in allow_rasterization.<locals>.draw_wrapper(artist,
renderer)
     69     if artist.get_agg_filter() is not None:
     70         renderer.start_filter()
---> 72     return draw(artist, renderer)
     73 finally:
     74     if artist.get_agg_filter() is not None:

File ~\AppData\Roaming\Python\Python312\site-packages\mpl_toolkits\
mplot3d\axis3d.py:649, in Axis.draw_grid(self, renderer)
    645     return
    647 renderer.open_group("grid3d", gid=self.get_gid())
--> 649 ticks = self._update_ticks()
    650 if len(ticks):
    651     # Get general axis information:
    652     info = self._axinfo

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
axis.py:1301, in Axis._update_ticks(self)
   1296 """
   1297 Update ticks (position and labels) using the current data
interval of
   1298 the axes.  Return the list of ticks that will be drawn.
   1299 """
   1300 major_locs = self.get_majorticklocs()
-> 1301 major_labels = self.major.formatter.format_ticks(major_locs)
   1302 major_ticks = self.get_major_ticks(len(major_locs))
```

```
   1303 for tick, loc, label in zip(major_ticks, major_locs,
major_labels):

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
ticker.py:216, in Formatter.format_ticks(self, values)
    214 def format_ticks(self, values):
    215     """Return the tick labels for all the ticks at once."""
--> 216     self.set_locs(values)
    217     return [self(value, i) for i, value in enumerate(values)]

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
ticker.py:728, in ScalarFormatter.set_locs(self, locs)
    726     self._compute_offset()
    727 self._set_order_of_magnitude()
--> 728 self._set_format()

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
ticker.py:821, in ScalarFormatter._set_format(self)
    818 if len(self.locs) < 2:
    819     # We needed the end points only for the loc_range
calculation.
    820     locs = locs[:-2]
--> 821 loc_range_oom = int(math.floor(math.log10(loc_range)))
    822 # first estimate:
    823 sigfigs = max(0, 3 - loc_range_oom)

ValueError: cannot convert float NaN to integer

------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
File ~\AppData\Roaming\Python\Python312\site-packages\IPython\core\
formatters.py:402, in BaseFormatter.__call__(self, obj)
    400     pass
    401 else:
--> 402     return printer(obj)
    403 # Finally look for special method names
    404 method = get_real_method(obj, self.print_method)

File ~\AppData\Roaming\Python\Python312\site-packages\IPython\core\
pylabtools.py:170, in print_figure(fig, fmt, bbox_inches, base64,
**kwargs)
    167     from matplotlib.backend_bases import FigureCanvasBase
    168     FigureCanvasBase(fig)
--> 170 fig.canvas.print_figure(bytes_io, **kw)
    171 data = bytes_io.getvalue()
    172 if fmt == 'svg':

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
```

```
backend_bases.py:2175, in FigureCanvasBase.print_figure(self,
filename, dpi, facecolor, edgecolor, orientation, format, bbox_inches,
pad_inches, bbox_extra_artists, backend, **kwargs)
   2172     # we do this instead of
`self.figure.draw_without_rendering`
   2173     # so that we can inject the orientation
   2174     with getattr(renderer, "_draw_disabled", nullcontext)():
-> 2175         self.figure.draw(renderer)
   2176 if bbox_inches:
   2177     if bbox_inches == "tight":

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
artist.py:95, in _finalize_rasterization.<locals>.draw_wrapper(artist,
renderer, *args, **kwargs)
     93 @wraps(draw)
     94 def draw_wrapper(artist, renderer, *args, **kwargs):
---> 95     result = draw(artist, renderer, *args, **kwargs)
     96     if renderer._rasterizing:
     97         renderer.stop_rasterizing()

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
artist.py:72, in allow_rasterization.<locals>.draw_wrapper(artist,
renderer)
     69     if artist.get_agg_filter() is not None:
     70         renderer.start_filter()
---> 72     return draw(artist, renderer)
     73 finally:
     74     if artist.get_agg_filter() is not None:

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
figure.py:3162, in Figure.draw(self, renderer)
   3159             # ValueError can occur when resizing a window.
   3161     self.patch.draw(renderer)
-> 3162     mimage._draw_list_compositing_images(
   3163         renderer, self, artists, self.suppressComposite)
   3165     renderer.close_group('figure')
   3166 finally:

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
image.py:132, in _draw_list_compositing_images(renderer, parent,
artists, suppress_composite)
    130 if not_composite or not has_images:
    131     for a in artists:
--> 132         a.draw(renderer)
    133 else:
    134     # Composite any adjacent images together
    135     image_group = []

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
artist.py:72, in allow_rasterization.<locals>.draw_wrapper(artist,
```

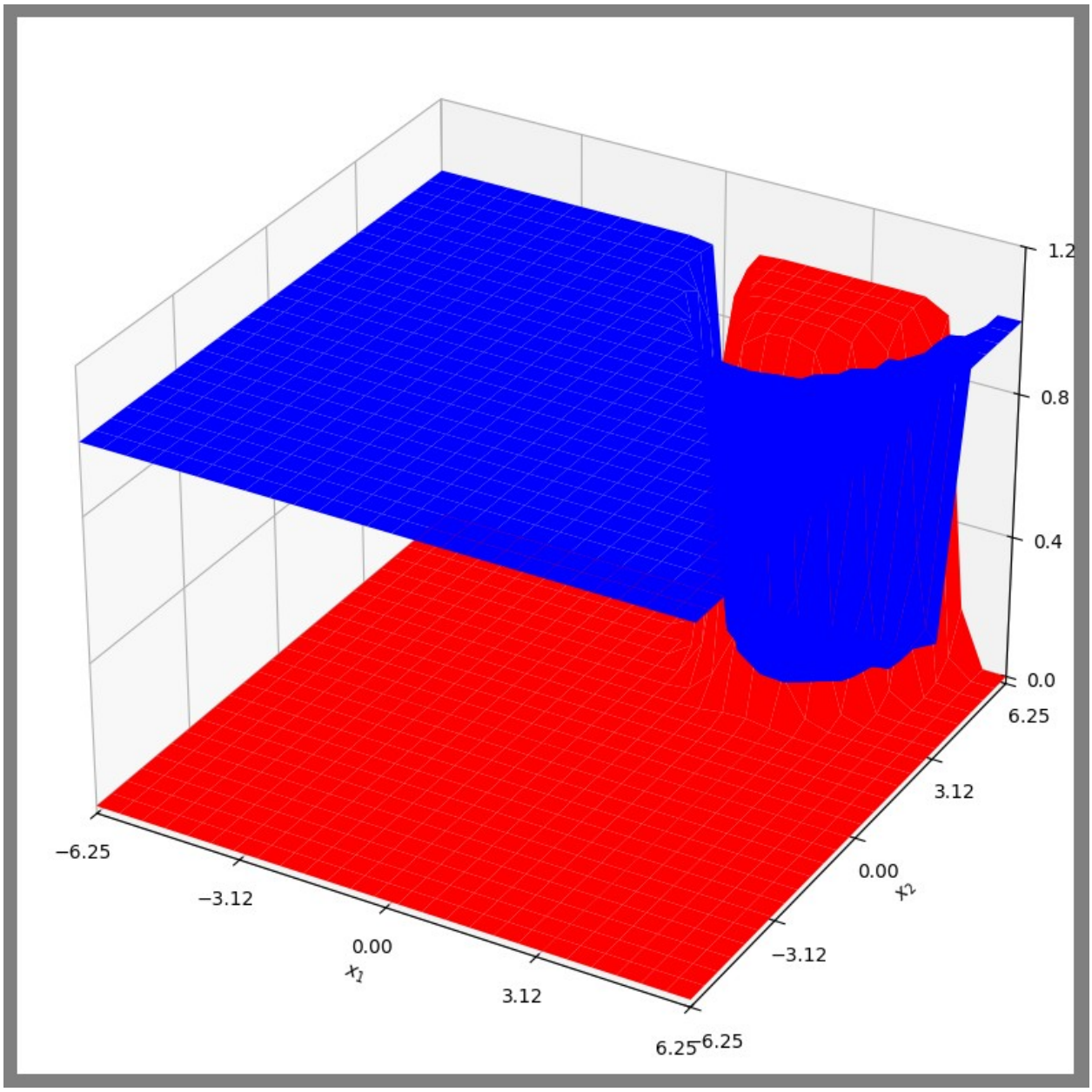
```
renderer)
     69     if artist.get_agg_filter() is not None:
     70         renderer.start_filter()
---> 72     return draw(artist, renderer)
     73 finally:
     74     if artist.get_agg_filter() is not None:

File ~\AppData\Roaming\Python\Python312\site-packages\mpl_toolkits\
mplot3d\axes3d.py:460, in Axes3D.draw(self, renderer)
    458 # Then gridlines
    459 for axis in self._axis_map.values():
--> 460     axis.draw_grid(renderer)
    461 # Then axes, labels, text, and ticks
    462 for axis in self._axis_map.values():

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
artist.py:72, in allow_rasterization.<locals>.draw_wrapper(artist,
renderer)
     69     if artist.get_agg_filter() is not None:
     70         renderer.start_filter()
---> 72     return draw(artist, renderer)
     73 finally:
     74     if artist.get_agg_filter() is not None:

File ~\AppData\Roaming\Python\Python312\site-packages\mpl_toolkits\
mplot3d\axis3d.py:649, in Axis.draw_grid(self, renderer)
    645     return
    647 renderer.open_group("grid3d", gid=self.get_gid())
--> 649 ticks = self._update_ticks()
    650 if len(ticks):
    651     # Get general axis information:
    652     info = self._axinfo

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
axis.py:1301, in Axis._update_ticks(self)
   1296 """
   1297 Update ticks (position and labels) using the current data
interval of
   1298 the axes.  Return the list of ticks that will be drawn.
   1299 """
   1300 major_locs = self.get_majorticklocs()
-> 1301 major_labels = self.major.formatter.format_ticks(major_locs)
   1302 major_ticks = self.get_major_ticks(len(major_locs))
   1303 for tick, loc, label in zip(major_ticks, major_locs,
major_labels):

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
ticker.py:216, in Formatter.format_ticks(self, values)
    214 def format_ticks(self, values):
    215     """Return the tick labels for all the ticks at once."""
```

```
--> 216     self.set_locs(values)
    217     return [self(value, i) for i, value in enumerate(values)]

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
ticker.py:728, in ScalarFormatter.set_locs(self, locs)
    726     self._compute_offset()
    727 self._set_order_of_magnitude()
--> 728 self._set_format()

File ~\AppData\Roaming\Python\Python312\site-packages\matplotlib\
ticker.py:821, in ScalarFormatter._set_format(self)
    818 if len(self.locs) < 2:
    819     # We needed the end points only for the loc_range
calculation.
    820     locs = locs[:-2]
--> 821 loc_range_oom = int(math.floor(math.log10(loc_range)))
    822 # first estimate:
    823 sigfigs = max(0, 3 - loc_range_oom)

ValueError: cannot convert float NaN to integer

<Figure size 1000x1000 with 1 Axes>
```

The KNN method expands V_n until R contains K_n samples, if we have less than K_n samples, then V_n tries to expand forever. In this case, K_n is not working for K_n=5 because we only have 4 samples for one class and 3 for the other.

Section k)

```
x1, x2, my, Sgm, posterior, df = labsol3()
classplot(posterior, x1, x2, 0, gsv={'gsv': 1, 'figstr': 'pdf'})
```



Authors

Daniel Linfon Ye Liu

Anton Maestre Gomez