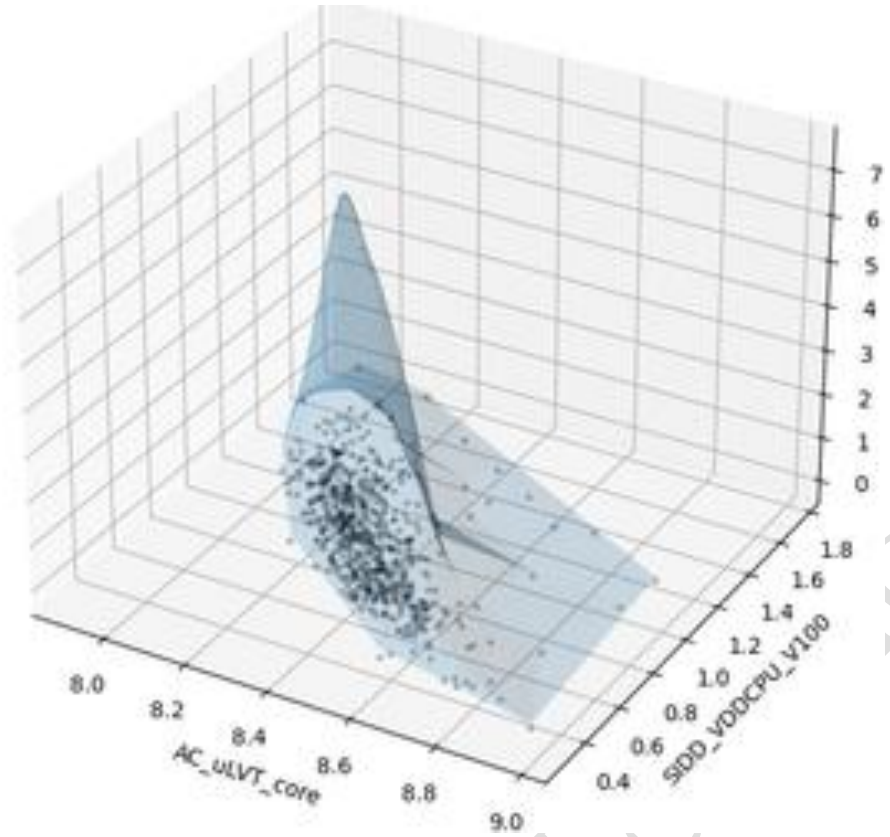
KDE and Data Outlier

Description	Attachment
ZX 6nm CP & FT data	03-FTH_DATA_0611.csv ZX_CP_demo.csv

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import gaussian_kde

path = <your_path>
dc = pd.read_csv(f'{path}/ZX_CP_demo.csv').set_index(['LWID']) # (64152, 5)

# KDE density with 1/10 subset
px,py = dc[['AC_uLVT_core','SIDD_VDDCPU_V100']].dropna().values[::100].T
kde = gaussian_kde(np.vstack([px, py]))
density = kde(np.vstack([px, py]))

plt.figure(figsize=(6,6))
ax1 = plt.subplot(111, projection='3d')
ax1.scatter(px,py,0, c='k', s=5, alpha=0.2)
ax1.plot_trisurf(px,py,density, alpha=0.2)
plt.tight_layout()
```

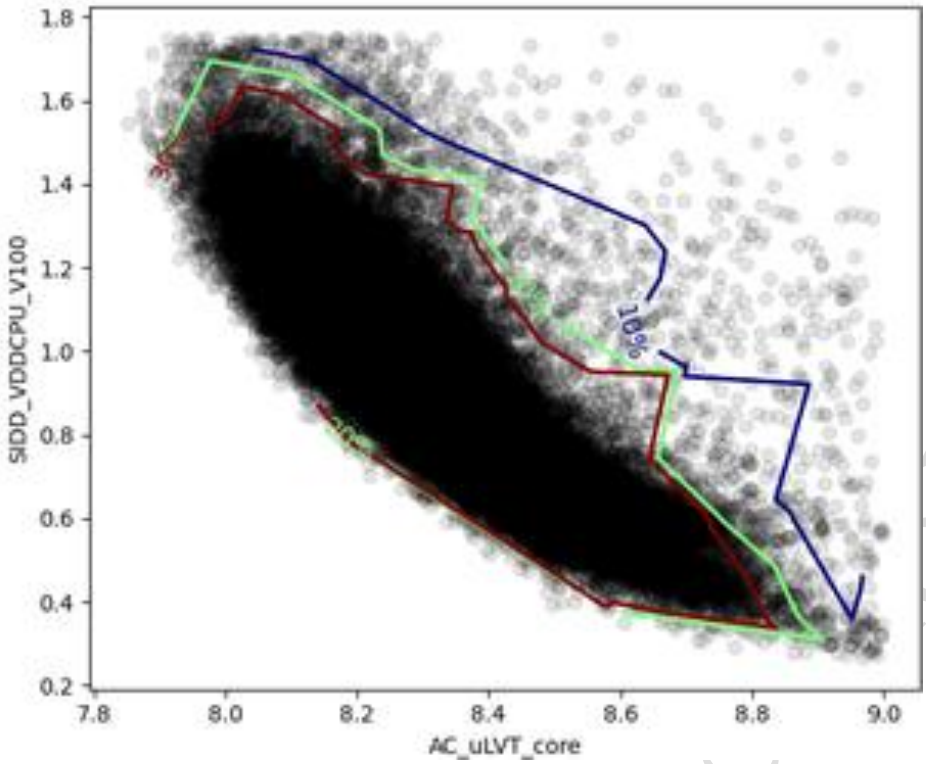
由於原始數據資料點有將近 65K, 上面例子我們以 1/100 採樣點 (以約 1K~10K 筆離散點即足夠) 建立高斯核函數密度, x 與 y 軸數據分別表徵 CPU 速度與 leakage 的特徵參數, AC_uLVT_core 與 SIDD_VDDCPU_V100.

若我們想將 KDE density 曲面 10% contour 邊界外圍的 samples 剔除, 怎麼做呢?



首先可以透過 tricontour 與自訂義的邊界 levels, 產生所有條件的決策邊界, e.g., [0.1, 0.2, 0.3] 分別對應 10%, 20% 與 30% contour 的多邊形 polygon, 並用 clabel 把標籤直接嵌入等高線. 由於 clabel 不支持 3D 作圖所以我們以 2D scatter 來呈現.

```
levels = [0.1, 0.2, 0.3]
clevel = {v:f'{v*100:.0f}%' for v in levels}

plt.figure(figsize=(6,5))
ax2 = plt.subplot(111)
contour = ax2.tricontour(px, py, density, cmap='jet', levels=levels,
linewidths=2, zorder=2)
ax2.clabel(contour, fontsize=12, fmt=clevel, zorder=3)
ax2.scatter(*dc[['AC_uLVT_core','SIDD_VDDCPU_V100']].dropna().values.T,c='k',a
lpha=0.1)
ax2.set_xlabel('AC_uLVT_core')
ax2.set_ylabel('SIDD_VDDCPU_V100')
plt.tight_layout()
```



代表決策邊界的 polygon 會記錄在 contour.collections, 依據前項 levels 定義, 所回傳 collections 第一筆 collections[0]表 10%, collections[1] 表 20%, 依此類推. 有時候回傳邊界以 piece-wise 多邊形組合, 我們將其全部收集並攤開成單一 polygon.

```
# collect 10% contour vertices
paths = contour.collections[0].get_paths() # 10%
vp = [] # collect piecewise vertices, when len(paths)>1
for i, path in enumerate(paths):
    vertices = path.vertices
    vp += [vertices]
vp = np.vstack(vp)
```

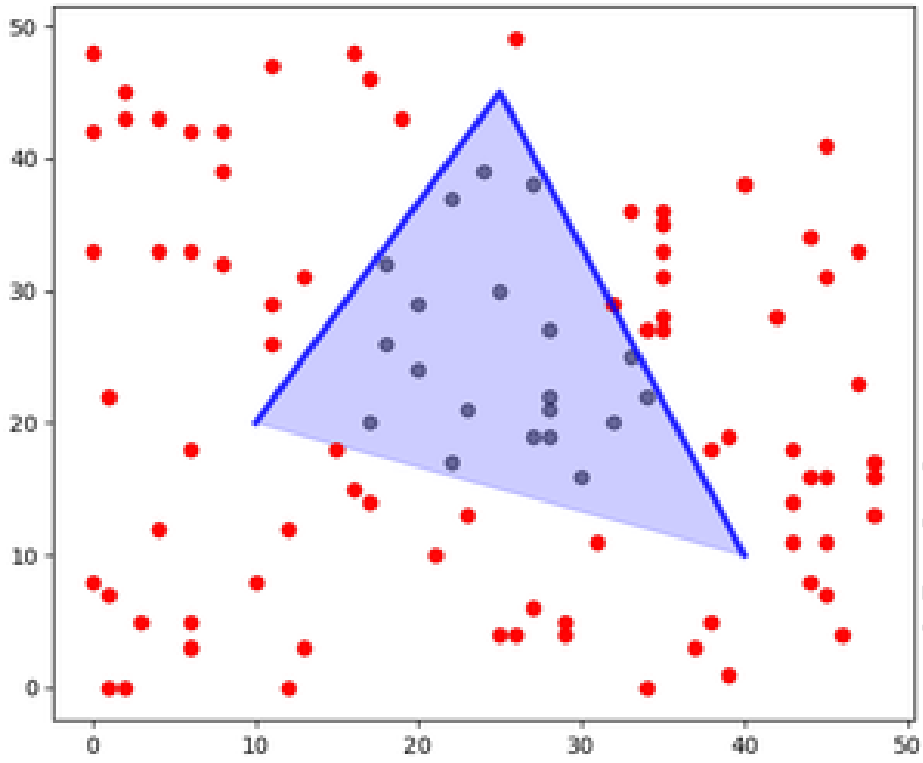
接下來標註 10% contour 邊界外的資料點. 判斷一群 2D (x, y) 資料點 points 是否在一組 polygon 範圍內最簡單的作法是直接調用 matplotlib.path.Path 物件.

```
contour_path = plt.matplotlib.path.Path(vp)
mask = np.full(points.shape[0], False)
mask = ~contour_path.contains_points(points)
```

但為了能讓 CP 工程人員在實際量產工作能夠處理, 我們也能提供以右射線法則的函示庫工具達成.

```
#%% Ray method tests whether a point is within a polygon
def isInPolygon(x,y,polygon):
    n = len(polygon)
    inside = 0
    for i in range(n):
        x1,y1 = polygon[i%n]
        x2,y2 = polygon[(i+1)%n]
        if min(y1,y2) < y <= max(y1,y2):
            if x <= max(x1,x2):
                if y1 != y2:
                    xc = (y-y2)*(x1-x2)/(y1-y2)+x2
                if x1 == x2 or x <= xc:
                    inside = not inside # cross number
    return inside

points = np.random.randint(0,50,(100,2))
polygon = np.array([[10,20],[25,45],[40,10]])

mask = np.array([False]*points.shape[0])
for i,(x,y) in enumerate(points):
    mask[i] = not isInPolygon(x,y,polygon)

plt.figure(figsize=(6,6))
plt.plot(*polygon.T, lw=3, c='b', alpha=0.8)
plt.fill(*polygon.T, c='b', alpha=0.2)
plt.scatter(*points.T, c='k', alpha=0.5, zorder=0)
plt.scatter(*points[mask].T, c='r')
plt.tight_layout()
```

我們隨機產生 [0, 50] 的 (x, y) 座標點 100 筆, 並判斷哪些在 polygon 外面並亮紅色.

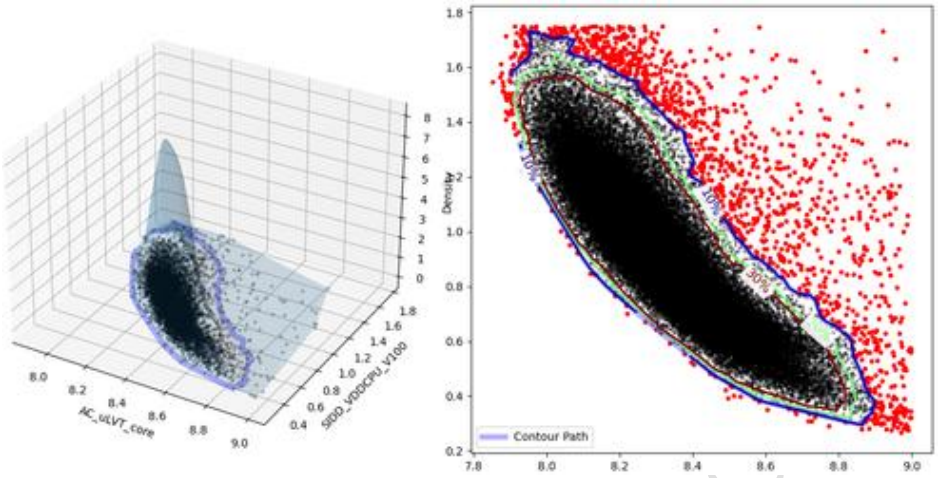


結合上面技巧, 先以 1/1000 sub samples 建立高斯 KDE, 透過 tricontour 取得決策邊界, 帶入全部資料點, 再 以上述兩種技巧之一將 10% contour polygon 之外的點標註紅色.

```
#%% CP KDE density and outlier analysis
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
#from scipy.spatial import ConvexHull
from scipy.stats import gaussian_kde

path = <your_path>
dc = pd.read_csv(f'{path}/ZX_CP_demo.csv').set_index(['LWID']) # (64152, 5)

# KDE density with 1/10 subset
px,py = dc[['AC_uLVT_core','SIDD_VDDCPU_V100']].dropna().values[::10].T
kde = gaussian_kde(np.vstack([px, py]))
density = kde(np.vstack([px, py]))

plt.figure(figsize=(12,6))
ax1 = plt.subplot(121, projection='3d')
ax1.scatter(px,py,0, c='k', s=5, alpha=0.2)
ax1.plot_trisurf(px,py,density, alpha=0.2)
```

```
ax1.set_xlabel('AC_uLVT_core')
ax1.set_ylabel('SIDD_VDDCPU_V100')
ax1.set_zlabel('Density')

# clabel is not yet implemented for 3D axes
levels = [0.1, 0.2, 0.3]
clevel = {v:f'{v*100:.0f}%' for v in levels}
ax2 = plt.subplot(122)
contour = ax2.tricontour(px, py, density, cmap='jet', levels=levels,
linewidths=2, zorder=2)
cl = ax2.clabel(contour, fontsize=12, fmt=clevel, zorder=3)
[txt.set_bbox(dict(facecolor='white', edgecolor='none', pad=2, alpha=0.9)) for
txt in cl]

# collect 10% contour vertices
paths = contour.collections[0].get_paths() # 10%
vp = [] # collect piecewise vertices, when len(paths)>1
for i, path in enumerate(paths):
    vertices = path.vertices
    vp += [vertices]
vp = np.vstack(vp)

# test full set
px,py = dc[['AC_uLVT_core','SIDD_VDDCPU_V100']].dropna().values.T
points = np.vstack((px, py)).T

# determine whether a point is within a polygon
if False: # (1) with matplotlib.path.Path
    contour_path = plt.matplotlib.path.Path(vp)
    outside_mask = np.full(points.shape[0], False)
    outside_mask = ~contour_path.contains_points(points)

if True: # (2) right ray method
    #hull = ConvexHull(vp)
    #hvertices = vp[hull.vertices]
    outside_mask = np.array([False]*points.shape[0])
    for i,(x,y) in enumerate(points):
        outside_mask[i] = not isInPolygon(x,y,vp)

x_outside = px[outside_mask]
y_outside = py[outside_mask]
```

```
ax2.scatter(px,py,s=5,c='k',alpha=0.3, zorder=0)
ax2.scatter(x_outside,y_outside,s=10,c='r',alpha=1)
ax2.plot(*vp.T, 'b-', lw=5, alpha=0.3, label='Contour Path')
ax1.plot(*vp.T, 'b-', lw=5, alpha=0.3) # clabel is not yet implemented for 3D
axes
ax2.legend()
plt.tight_layout()
```



前面的範例有兩個嚴重的問題, 第一個是當 x, y 的特徵向量值域是離散且明顯數量及差異時, 採用原始數據點建立的 KDE contour 會有斷層. 此時應該採用內插補點來提升解析度並計算相較均勻的網格.

```
x,y = d.values.T

# intepolate KDE with suset
sub = int(d.shape[0]/5000)
kde = gaussian_kde(np.vstack([x[::sub], y[::sub]]))
x1,x2 = x.min(),x.max()
y1,y2 = y.min(),y.max()
xi,yi = np.mgrid[x1:x2:100j, y1:y2:100j]
zi = kde(np.vstack([xi.ravel(), yi.ravel()]))
```

第二個問題是, 我們關心的是 CDF 10% (Yield), 而不是 KDE 值域的 10%, 此時需要一點技巧得到 CDF 對應 KDE 的值, 再把我們真正關心的 CDF (Yield) Contour 與相對應的 KDE level 位置以 CDF 刻度標出來.

```
# build CDF
S = np.sort(zi) # ranked density
cdf = np.cumsum(S)*100/zi.sum()
levels = np.interp(percentiles, cdf, S)  # map % to density
```

```
ctags = {l:f'{p}%' for l,p in zip(levels,percentiles)} # contour tags
```

把所有修正整理成一個函式呼叫.

```
def
cdfContour(df,fx='Vcore_3000M_8C',fy='Pcore_3000M_8C',percentiles=[5,50,95],su
rface=False,drop=False):
    '''sequence of percentiles between 0 and 100 inclusive,
       the 1st percentile will be used as the outlier boundary'''
    dt = df[[fx,fy]].dropna()
    q = dt.quantile(q=[0.0015,0.9985]) # drop 1/1000
    d = dt[np.all(dt<=q.loc[0.9985],axis=1)]
    m = d.mean()
    x,y = d.values.T
    print(f'drop {df.shape[0]-dt.shape[0]} nan samples')
    print(f'{d.shape[0]*100/dt.shape[0]:.2f} % yield for analysis')

    # intepolate KDE with suset
    sub = np.ceil(d.shape[0]/5000).astype(int)
    kde = gaussian_kde(np.vstack([x[::sub], y[::sub]]))
    x1,x2 = x.min(),x.max()
    y1,y2 = y.min(),y.max()
    xi,yi = np.mgrid[x1:x2:100j, y1:y2:100j]
    zi = kde(np.vstack([xi.ravel(), yi.ravel()]))

    # build CDF
    S = np.sort(zi) # ranked density
    cdf = np.cumsum(S)*100/zi.sum()
    levels = np.interp(percentiles, cdf, S)  # map % to density
    ctags = {l:f'{p}%' for l,p in zip(levels,percentiles)} # contour tags

    pnum = 2 if surface else 1
    fig = plt.figure(figsize=(6*pnum,6))
    ax2 = plt.subplot(1,pnum,pnum,title=f'CDF,
{d.shape[0]*100/dt.shape[0]:.2f} % yield for analysis')
    ax2.scatter(x,y, s=5, alpha=0.3, c='k', zorder=0)

ax2.scatter(*m.values.T,s=100,marker='^',c='r',label=f'$\mu$={m.values.round(3
)}')
    cs = ax2.contour(xi, yi,
zi.reshape(xi.shape),levels=levels,linewidths=3,cmap='jet', alpha=0.8)
```

```
    cl = plt.clabel(cs, inline=True, fontsize=12, fmt=ctags) # not inplemented
in 3D
    [txt.set_bbox(dict(facecolor='white', edgecolor='none', pad=1, alpha=0.9))
for txt in cl]
    ax2.grid(which='major',linestyle='-',alpha=0.5)
    ax2.grid(which='minor',linestyle=':',alpha=0.5)
    ax2.minorticks_on()
    ax2.set_xlabel(fx)
    ax2.set_ylabel(fy)
    ax2.legend()
    plt.tight_layout()

    if surface:
        ax1 = plt.subplot(1,pnum,1,projection='3d')
        ax1.set_title(f'Probability Density {d.shape[0]:,} smaples',y=1)
        ax1.scatter(x,y,0, s=5, alpha=0.3, c='k', zorder=0)
        ax1.plot_trisurf(xi.ravel(),yi.ravel(),zi,cmap='Blues',alpha=0.6,
zorder=1)
        ax1.tricontour(xi.ravel(), yi.ravel(), zi, levels=levels,
linewidths=3, cmap='jet',alpha=1,zorder=2)
        ax1.plot_surface(xi*0+x.max()*1.01, yi, zi.reshape(xi.shape),
alpha=0.1, color='g') # project x
        ax1.plot_surface(xi, yi*0+y.max()*1.01, zi.reshape(xi.shape),
alpha=0.1, color='g') # project y
        ax1.set_box_aspect(None, zoom=1.2)
        ax1.set_xlabel(fx)
        ax1.set_ylabel(fy)
        ax1.set_box_aspect([1, 1, 1])
        ax1.set_zlim(zmin=0)
        ax1.set_zticks(np.interp(np.arange(0,101,20), cdf, S),minor=False)
        ax1.set_zticklabels(np.arange(0,101,20))
        fig.subplots_adjust(left=-0.02,right=0.98,top=0.95,wspace=0.1,
hspace=0)

    if drop:# 1st contour as the threshold
        paths = cs.collections[0].get_paths()
        vp = [] # collect piecewise vertices, when len(paths)>1
        for i, path in enumerate(paths):
            vertices = path.vertices
            vp += [vertices]
        vp = np.vstack(vp)
```

```
        contour_path = plt.matplotlib.path.Path(vp)
        mask = ~contour_path.contains_points(d.values)
        ax2.scatter(*d[mask].values.T,c='r',alpha=0.2,label=f'outliers:
{mask.sum():,}')
        ax2.legend()
        idx = sorted(set(df.index)-set(d[~mask].index))
        df = df.drop(index=idx)
    return df

dd =
cdfContour(dfh,fx='Vcore_3000M_8C',fy='Pcore_3000M_8C',percentiles=[1],surface
=True,drop=True) # drop 1%
dd =
cdfContour(dd,fx='Vcore_3000M_8C',fy='Pcore_3000M_8C',percentiles=[5,10,50],su
rface=True)
```

第一次呼叫函式標示 1% Yield Contour, 並將 outlier 剃除, 再呼叫函式第二次把 5%, 10%, 50% Yield 標示出來.