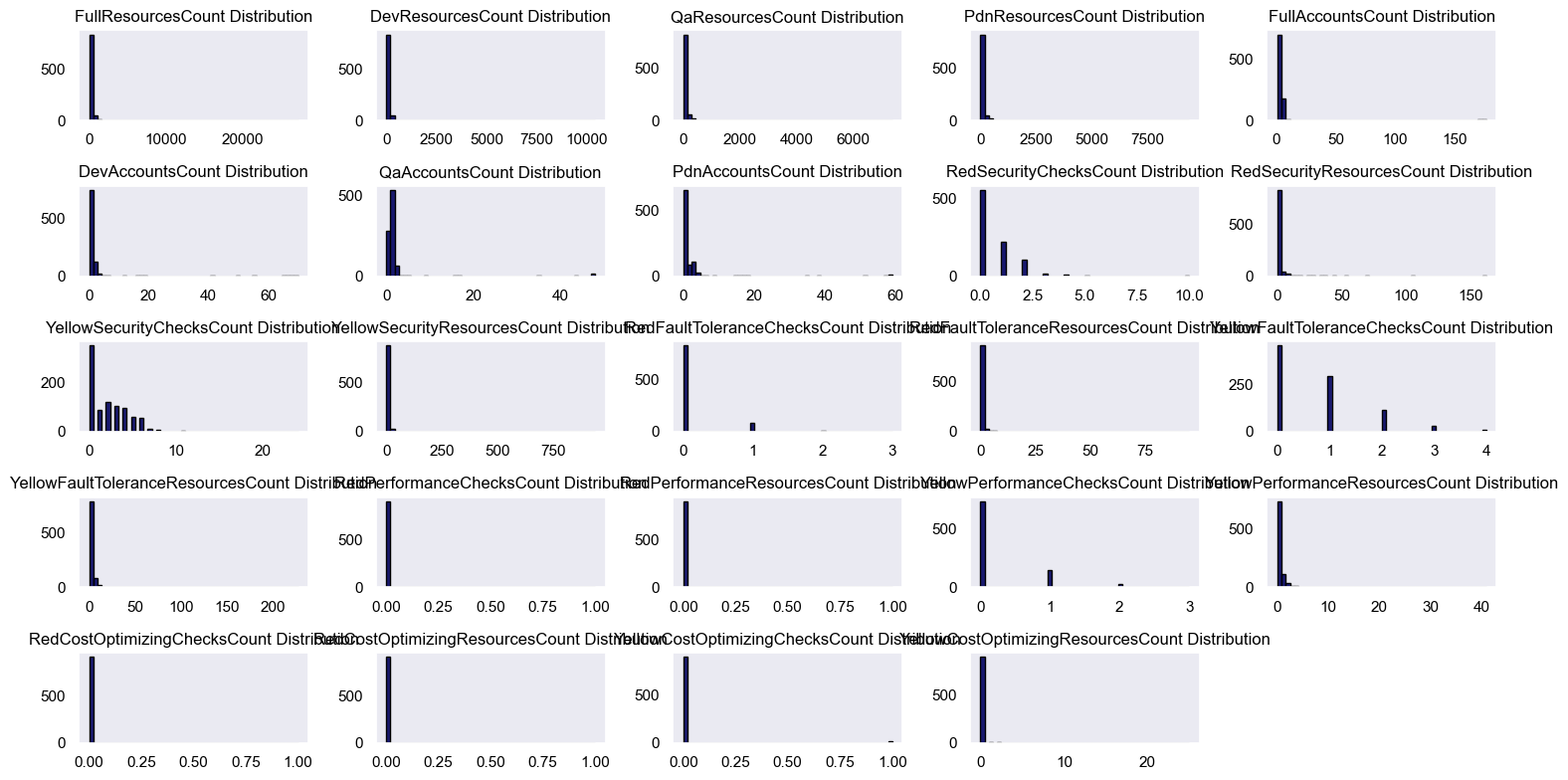
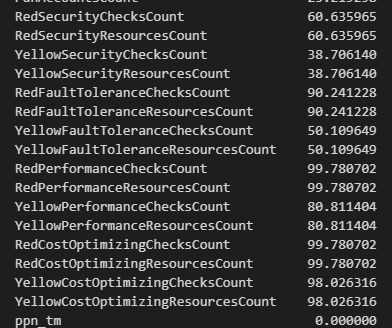
# Analísis de los datos

## Distribución de los datos



Se encuentra un alto porcentaje de datos en cero para cada variable. Mas que todo los relacionados con los checks. Esto significa que hay aplicaciones que no tienen checks reportados para algún pilar o en algún nivel de riesgo (Rojo, amarillo).



Para todas las variables asociadas a los checks (RedFault… YellowCost..) el percentil 75% es igual a cero.

Imagen que contiene Gráfico

Descripción generada automáticamente

# Preprocesamiento

Se hace eliminación de características que no aportan al modelo. En algunos casos tienen un valor fijo, son índices o categorías que pueden omitirse.

Eliminados: 'organization\_id', 'snapshot\_date\_id', 'application\_id', 'provider\_id', 'ppn\_tm'

# Implementación del Modelo

## Clusterización para pilar de Seguridad

Debido a la cantidad de ceros que se encuentran para cada pilar, como fue observado en el analísis inicial, se propone un análisis y procesamiento por cada pilar. Si se escoge un pilar pueden removerse las aplicaciones que no tienen checks para ese pilar, esta prueba se hace con el pilar Security.

Después de hacer el procesamiento, quedan 353 aplicaciones con checks de seguridad activos. En el histograma siguiente se ven las distribuciones de los datos.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

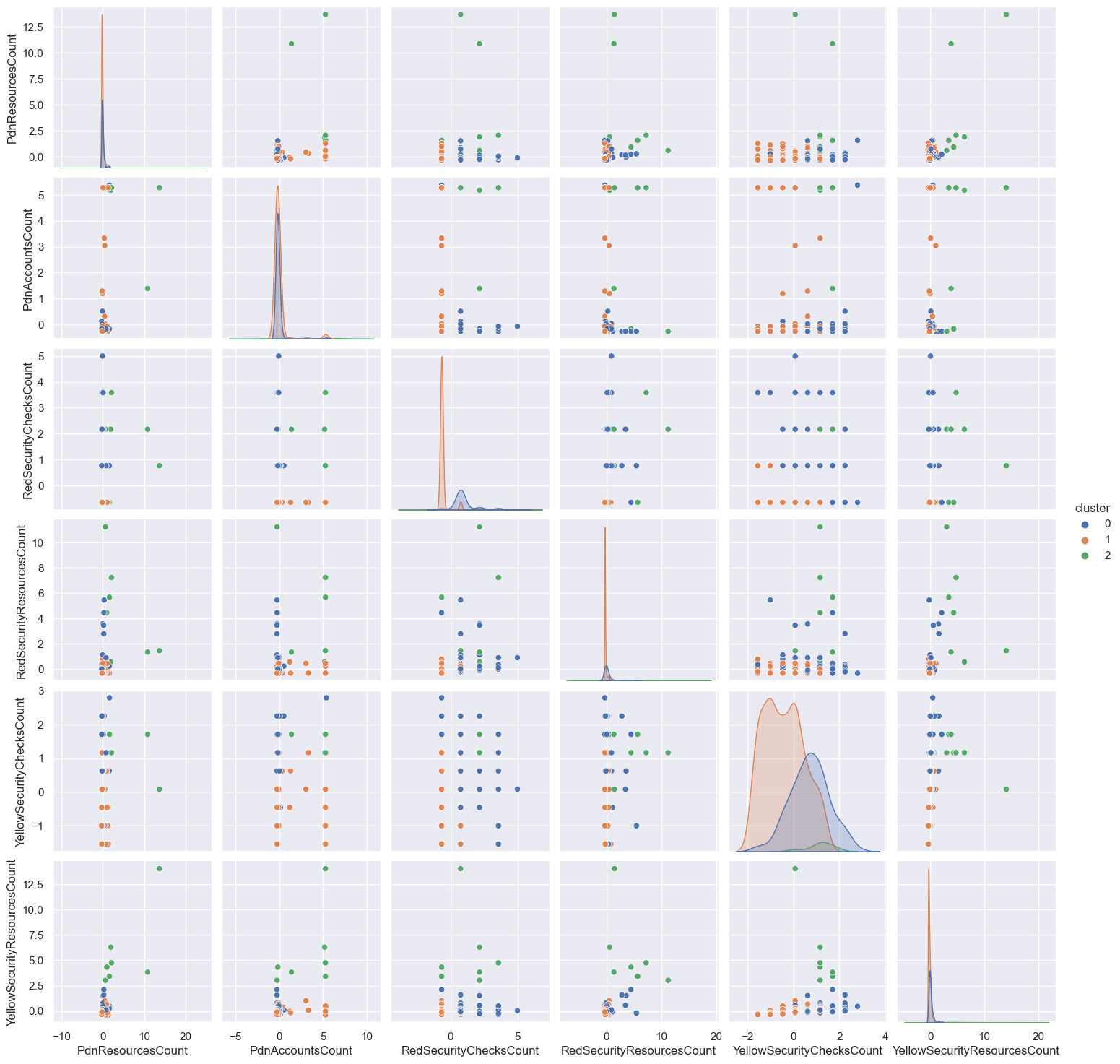
En la matriz de correlación, se puede observar algunas variables con una alta correlación, como PdnResourcesCount con YellowSecurityResourcesCount y RedSecurityResourcesCount. Esto es normal dado y puede interpretarse como que entre más recursos en una aplicación, más recursos marcados con checks en TrustedAdvisor. También hay una alta correlación entre YellowSecurityResourcesCount y RedSecurityResourcesCount, esto significa que entre más recursos tengan un check de un nivel de riesgo, más van a tener del otro tipo de riesgo, además el conteo de recursos será compartido, es decir, muchos recursos estarán marcados con un check rojo y amarillo, según el caso. Por ultimo se ve la poca correlación con las cuentas donde esta desplegada la aplicación, esto tiene sentido ya que es algo más propio de la arquitectura de despliegue en la aplicación, si es una aplicación transversal o si se ejecuta solo en determinadas cuentas (el 75% de PdnAccountsCount esta entre 1 y 3).

Imagen que contiene Gráfico

Descripción generada automáticamente

Con 3 clusters se encuentran 3 grupos que no están muy bien separados, se ve cierta incidencia por los valores atípicos, y algunos grupos se superponen. Hay un grupo marginal con 7 elementos, mientras los otros tienen entre 100 a 200 elementos. Surge la duda si el conteo de checks puede aportar, dado que es propio de los definidos por TrustedAdvisor. El servicio puede tener múltiples checks para un pilar y eso podría significar que hay más desviaciones de la infraestructura.

En general aquí se ven muy bien demarcados los grupos, y hay cierta facilidad para interpretarlos.



## Clusterización a variables Checks

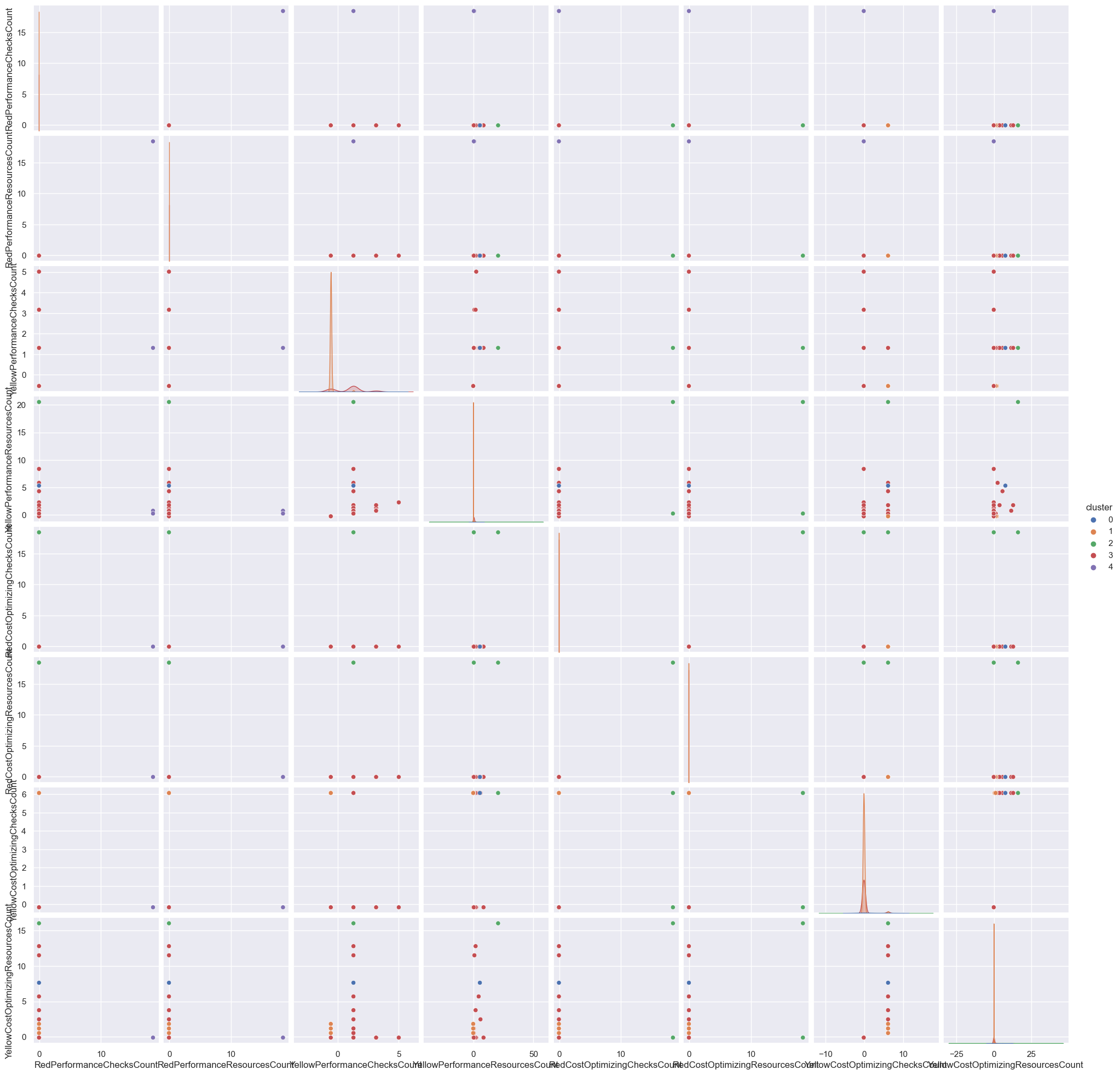
En este caso se realiza una clusterización con KMeans con k vecinos igual a 5. Se toman únicamente los datos de producción y se dejan como variables solo aquellas relacionados con los checks (conteo de checks por pilar por impacto, conteo de recursos por pilar por impacto).

Quedan en total 682 recursos después de realizar el filtro de producción. Para la normalización se utiliza la estandarización que deja los datos con una media de 0 y una desviación estándar de 1. Se deja la variable categórica applicationcode para poder asociar los registros.

Esta clusterización genera los siguientes grupos

|  |  |
| --- | --- |
| Cluster | Cantidad de registros |
| 0 | 9 |
| 1 | 436 |
| 2 | 2 |
| 3 | 233 |
| 4 | 2 |

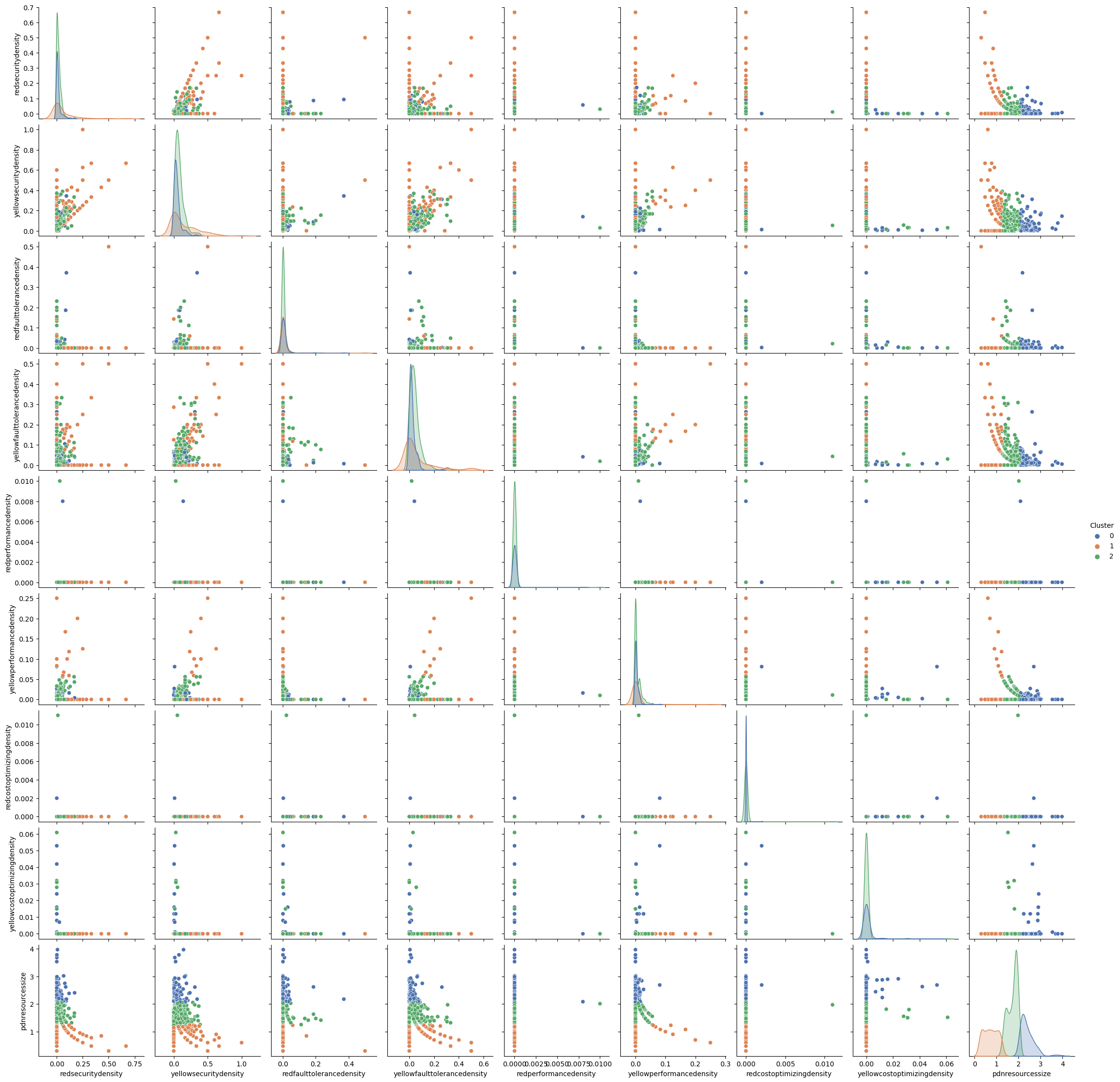
Nuevamente, como en la clusterización del pilar de seguridad, se evidencian 2 grupos dominantes, donde la mayoría de los datos son distribuidos. Una manera de solucionar es la eliminación de atípicos para evitar esos valores extremos que se quedan con pocos vecinos. También una medida de la separabilidad de los clusters puede servir para identificar si un clúster es óptimo.



## Densidad

### Clusterización con todos los checks

Aquí se dejaron todos los checks y la nueva variable pdnresourcessize



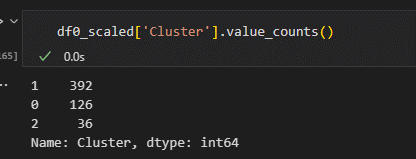
Gráfico, Gráfico de dispersión

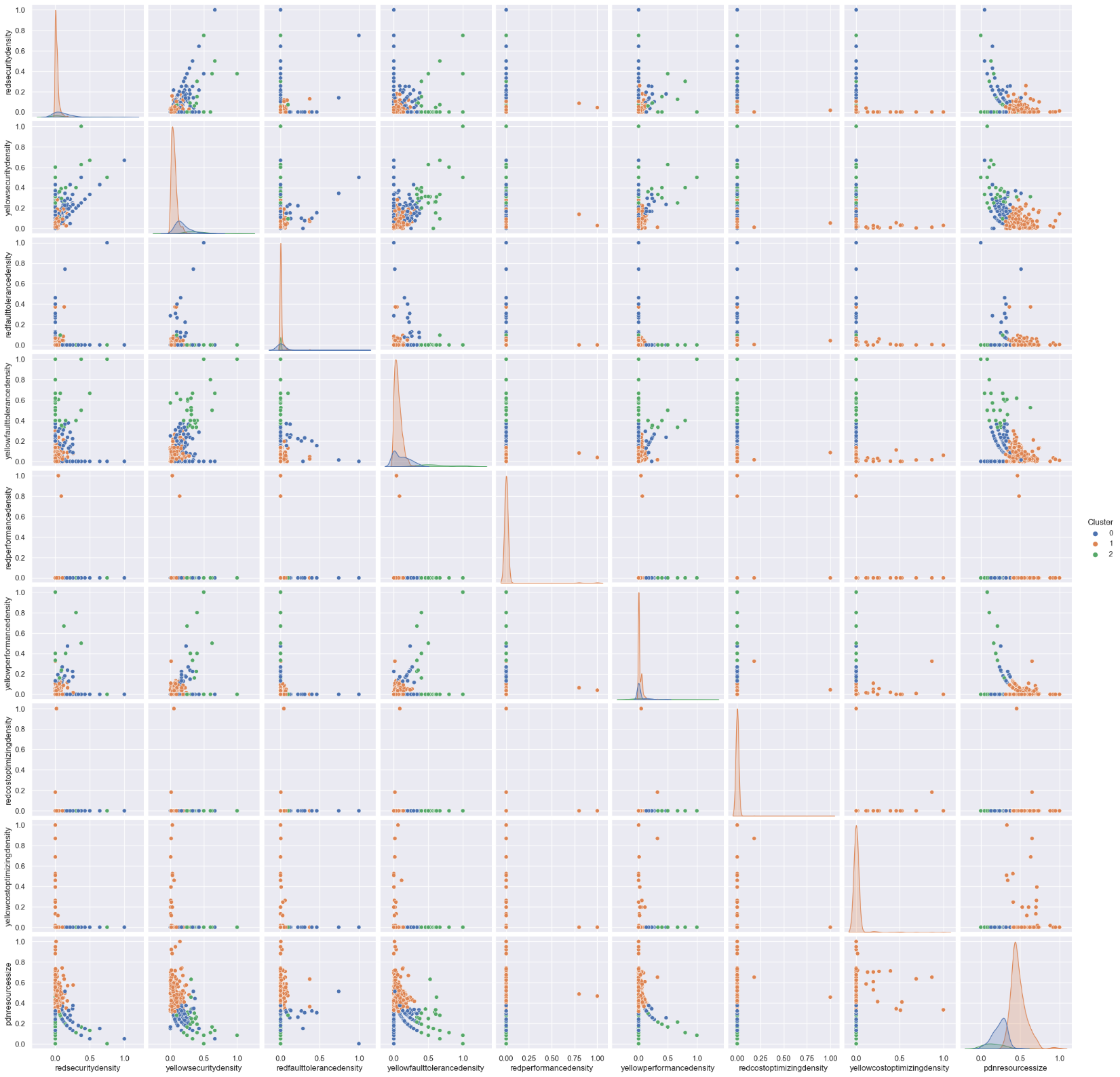
Descripción generada automáticamente

Se da una buena separación de clusters, pero se sigue observando algunos registros que no tienen checks para ninguno de los pilares que generan un ruido.

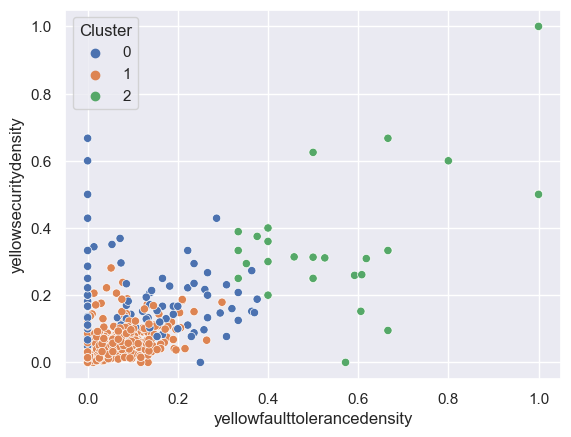
### Clusterización removiendo apps sin checks

Se eliminan las aplicaciones sin checks, y se deja la variable pdnresourcessize con normalización. Sería bueno hacer normalización. K=3



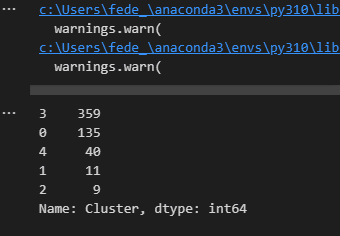


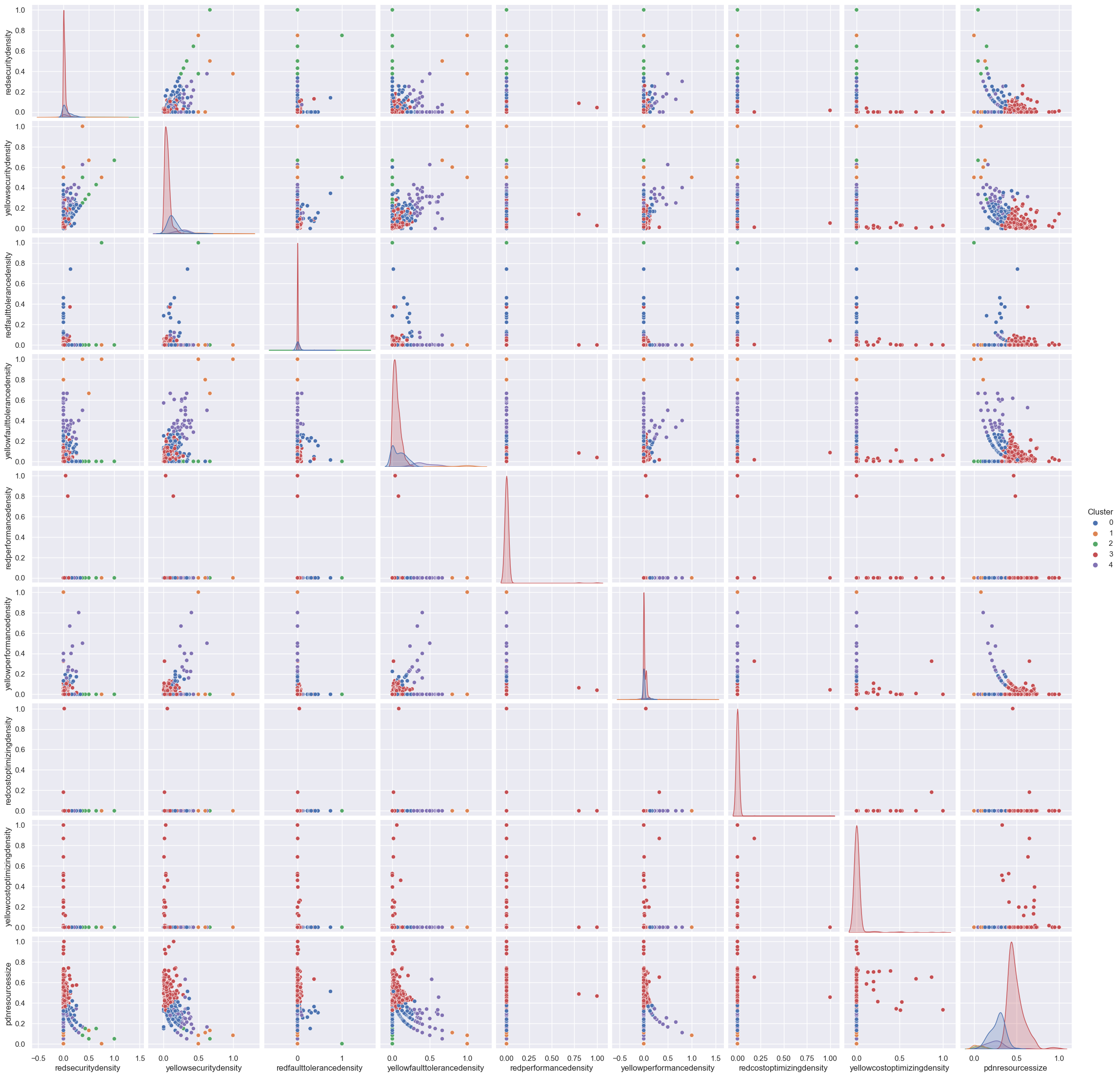
Relación entre dos variables de densidad: el grupo verde corresponde a aplicaciones con densidad de checks considerable

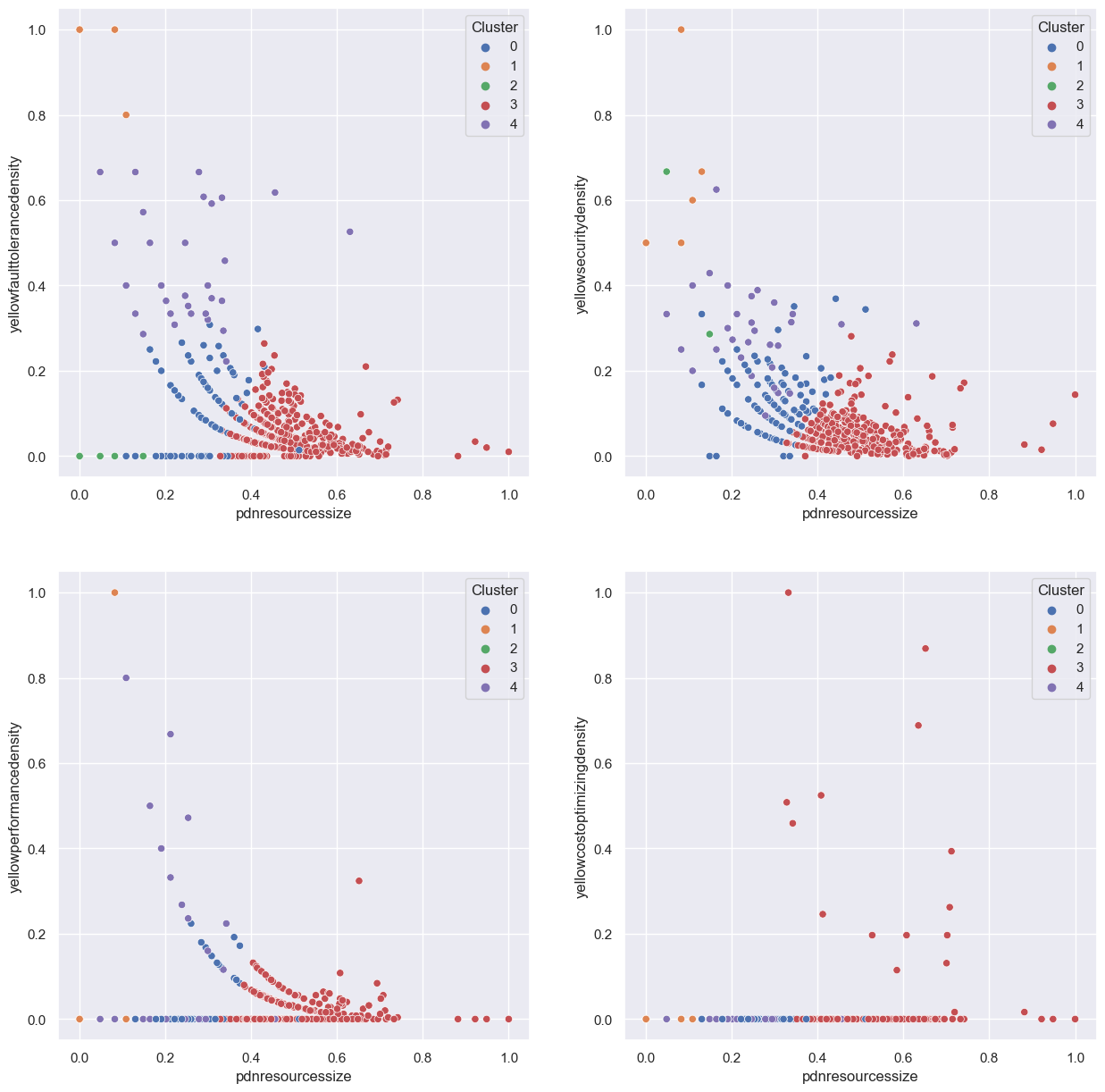


### Cluster kmeans k=5

Con normalización







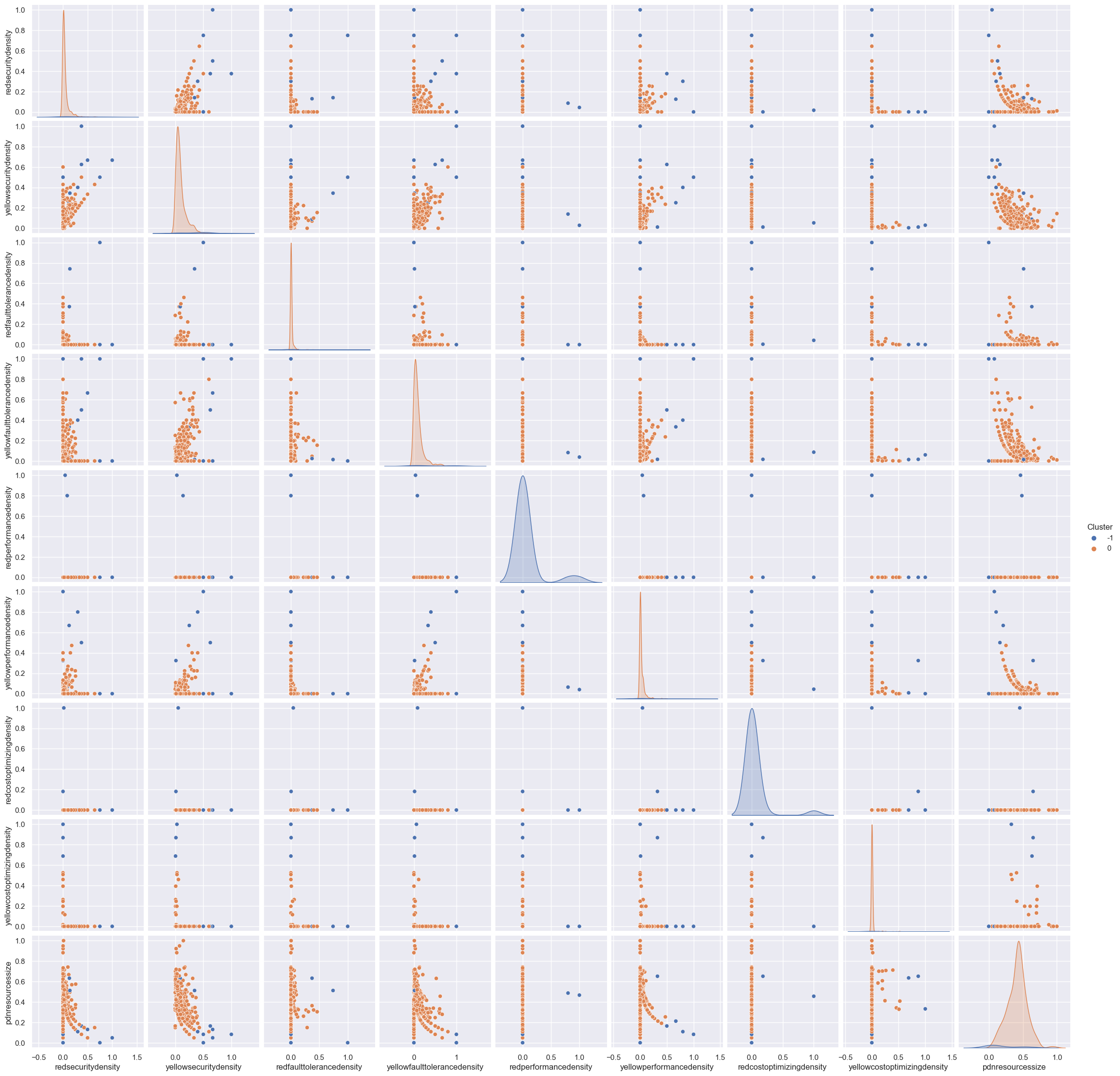
En general buenos resultados, solo se dificulta interpretar el pilar costoptimizing.

## DBSCAN

### Eps de 0.3, minpoints=11

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente con confianza media



Hay una buena distribución de los datos. Genera dos grupos. Se supera el problema del pilar de cost optimizing, quedan en azul las aplicaciones con alta densidad en costoptimizing, faulttolerance, performance y security. El resto de aplicaciones saldrían del grupo de las críticas.

Interfaz de usuario gráfica, Texto, Aplicación

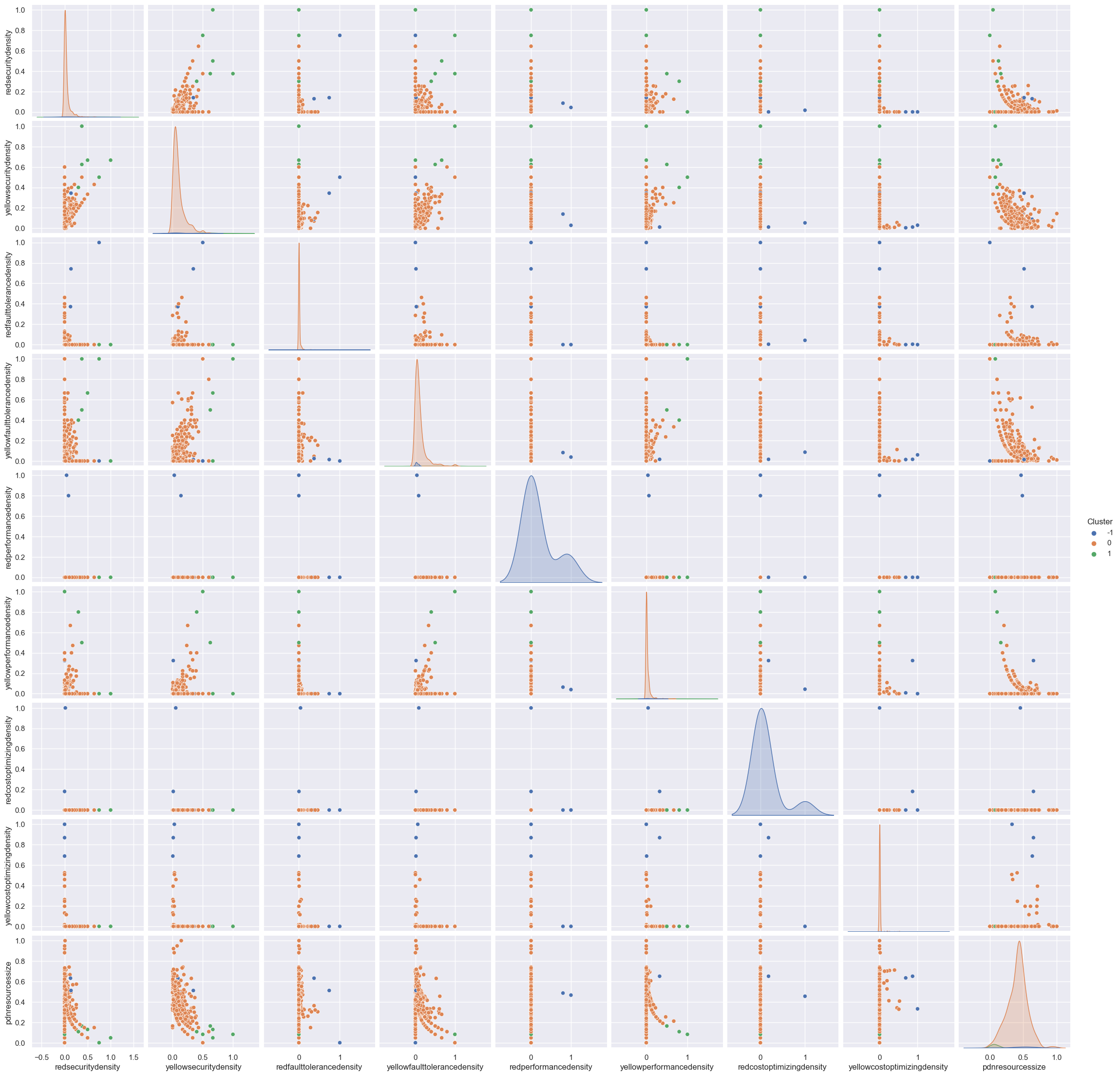
Descripción generada automáticamente

### Eps 0.9, min point=7

Se aumenta el radio para detectar vecinos, y se disminuye la cantidad de puntos mínimos por cluster

Texto

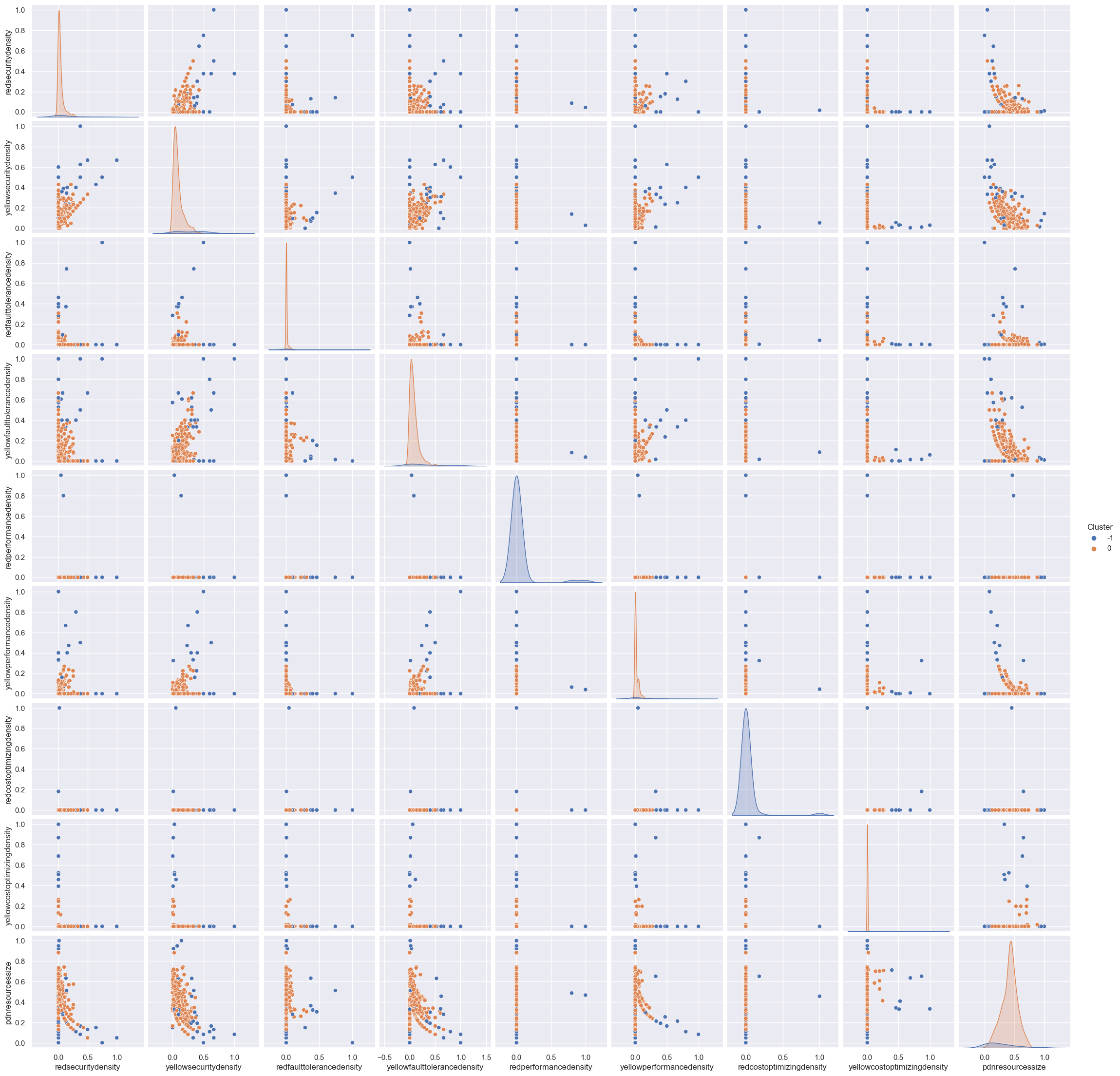
Descripción generada automáticamente con confianza baja



Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

### Eps 0.2, min points= 11



No hay muy buenos resultado, se contamina más el grupo azul con aplicaciones con baja densidad de checks.