graficas

April 6, 2022

1 OBTENCIÓN DE DATOS

[ ]:
```
import definirCorrelacionVariables
from matplotlib import pyplot
from sklearn.metrics import r2_score
from scipy.optimize import curve_fit
import pandas as pd
import seaborn as sns
import numpy as np
import pylab as pl

data= definirCorrelacionVariables.getDataFromDataBase()
```

2 pre PROCESAMIENTO DE DATA

[ ]:
```
dataframe,datos,dictGeneral = definirCorrelacionVariables.
  ↪generarMatrizDatos(data)
```

3. OBTENCIÓN DE CORRELACIÓN MÚLTIPLE

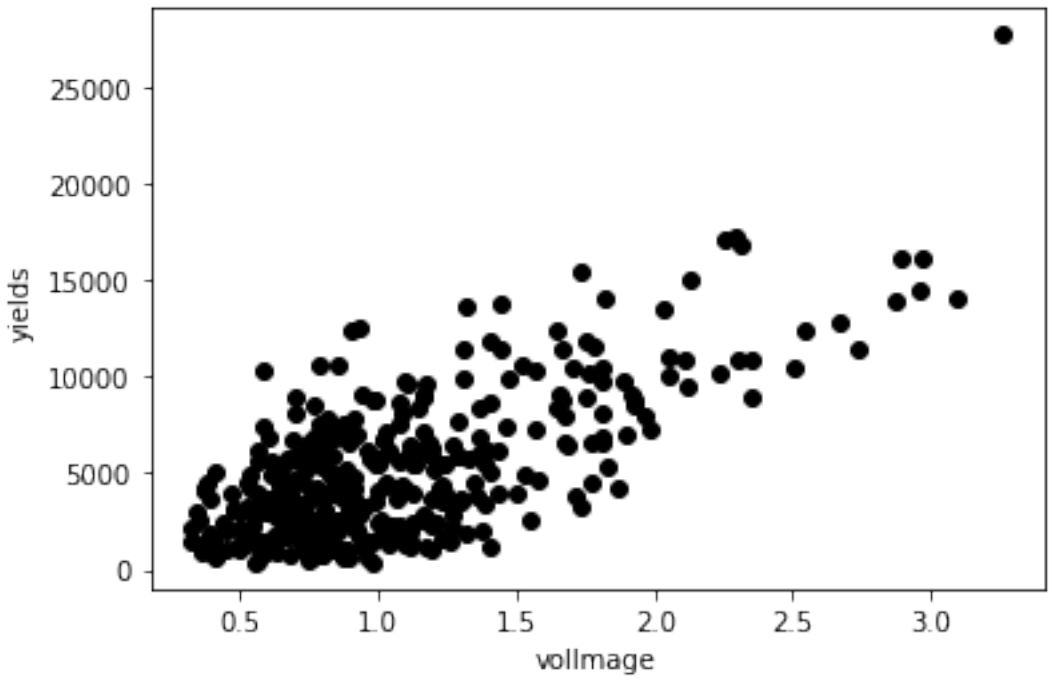
[ ]:
```
print(len(datos.datosYeld))
correlation=[]
correlation=dataframe.corr(method="pearson")
```

373

GRAFICAS ENTRE VARIABLES

[ ]:
```
#pyplot.scatter(datos.datosVolumen, datos.datosVolumenCalculado,c="red")
#pyplot.scatter(datos.datosArea, datos.datosAreaCalculada,c="red")
#pyplot.scatter(datos.datosNdvi, datos.datosVolumenCalculado,c="blue")
#pyplot.scatter(datos.datosAltura, datos.datosNdvi,c="green")
pyplot.scatter(dataframe["volImage"],dataframe["yields"],c="black")
pyplot.xlabel("volImage")
pyplot.ylabel("yields")

pyplot.show()
```



LAI CALCULATED FROM EQUATION BASED ON VOLUME and AREA

Se calcula el IAF usando Volumen Imagen y Yield

$IAF = 0.0134 + 2.7791 Vc$

Se calcula IAF usando Area Lateral y Yield

$IAF = -0.5786 + 0.7896 Alat $

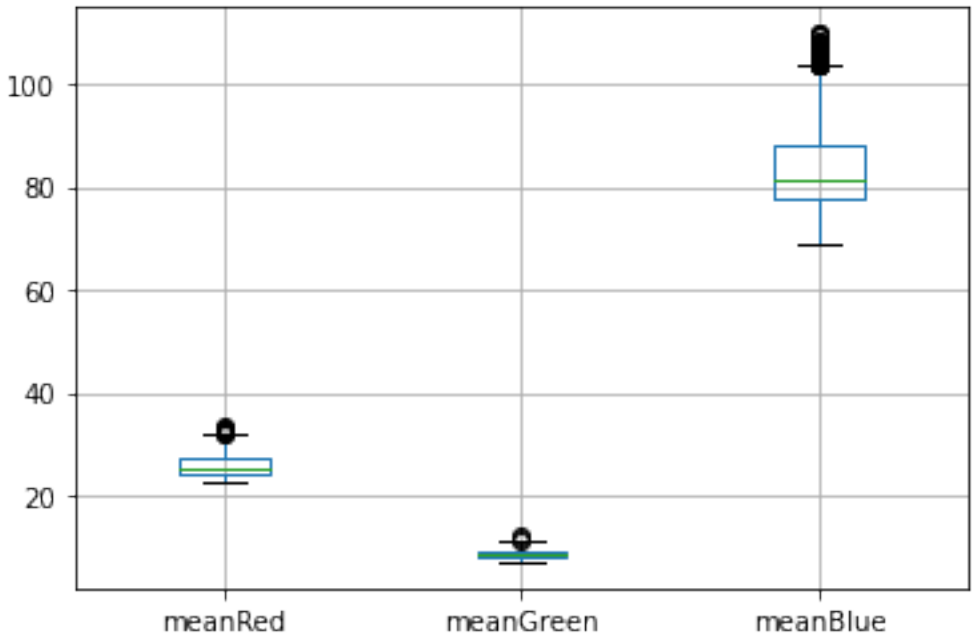
[ ]:
```
from sklearn.preprocessing import StandardScaler

import math
areaLateral = []
volumeFromDiameter = []
diametros=[]
for index,x in enumerate(datos.datosArea):
    diametro = 2*math.sqrt(x/math.pi)
    diametros.append(diametro)
    altura= datos.datosAlturaCalculada[index]/100
    valueAreaLateral=diametro*math.sqrt((diametro*diametro)+4*(altura*altura))
    valueVolume = math.pi*diametro*diametro*altura*(1/6)
    volumeFromDiameter.append(valueVolume)
    areaLateral.append(valueAreaLateral)
IAF_from_volume = definirCorrelacionVariables.objective(datos.
 ↪datosVolumenImagen, 2.7791,0.0134)
```

```
IAF_from_area = definirCorrelacionVariables.objective(areaLateral, 0.7896,-0.
↪5786)
d = {"latArea":areaLateral, "IAF_VOL":IAF_from_volume, "IAF_AREA":IAF_from_area}
dictGeneral.update(d)
dframeFinal = pd.DataFrame(data=dictGeneral)

scaled_features = StandardScaler().fit_transform(dframeFinal.values)
scaled_features_df = pd.DataFrame(scaled_features, index=dframeFinal.index,␣
↪columns=dframeFinal.columns)
dframeFinal.boxplot(column=['meanRed', 'meanGreen','meanBlue'])
```

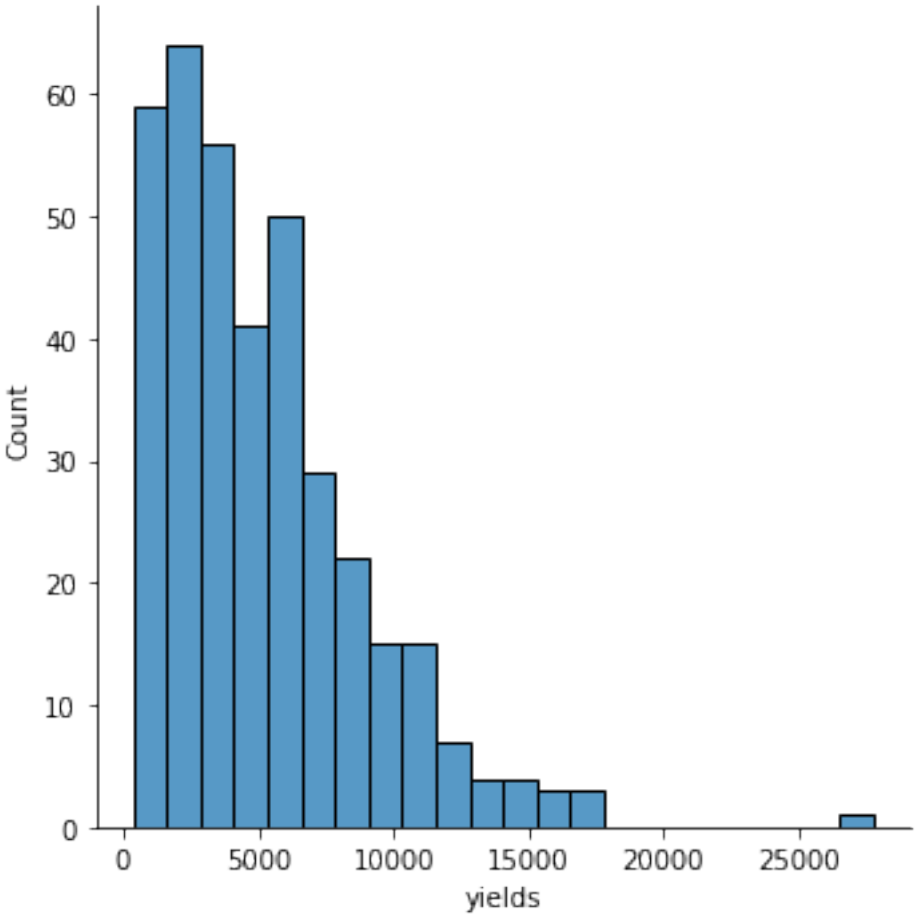
[ ]: <AxesSubplot:>



[ ]:
```
print(dframeFinal['yields'].kurt())
print(dframeFinal['yields'].skew())
sns.displot(dframeFinal['yields'])
```

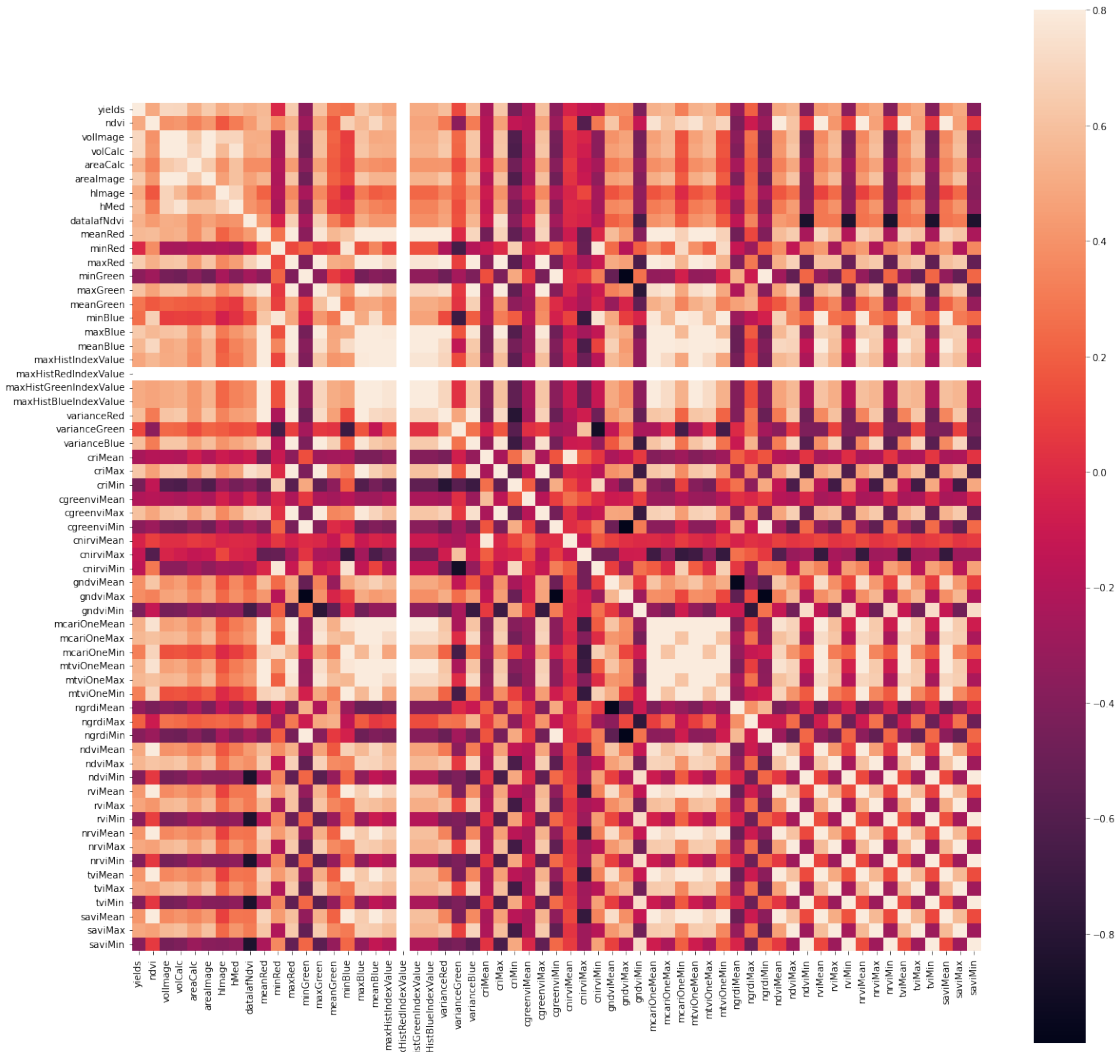
```
3.648259646881603
1.451541527339079
```

[ ]: <seaborn.axisgrid.FacetGrid at 0x1fe2199cf88>

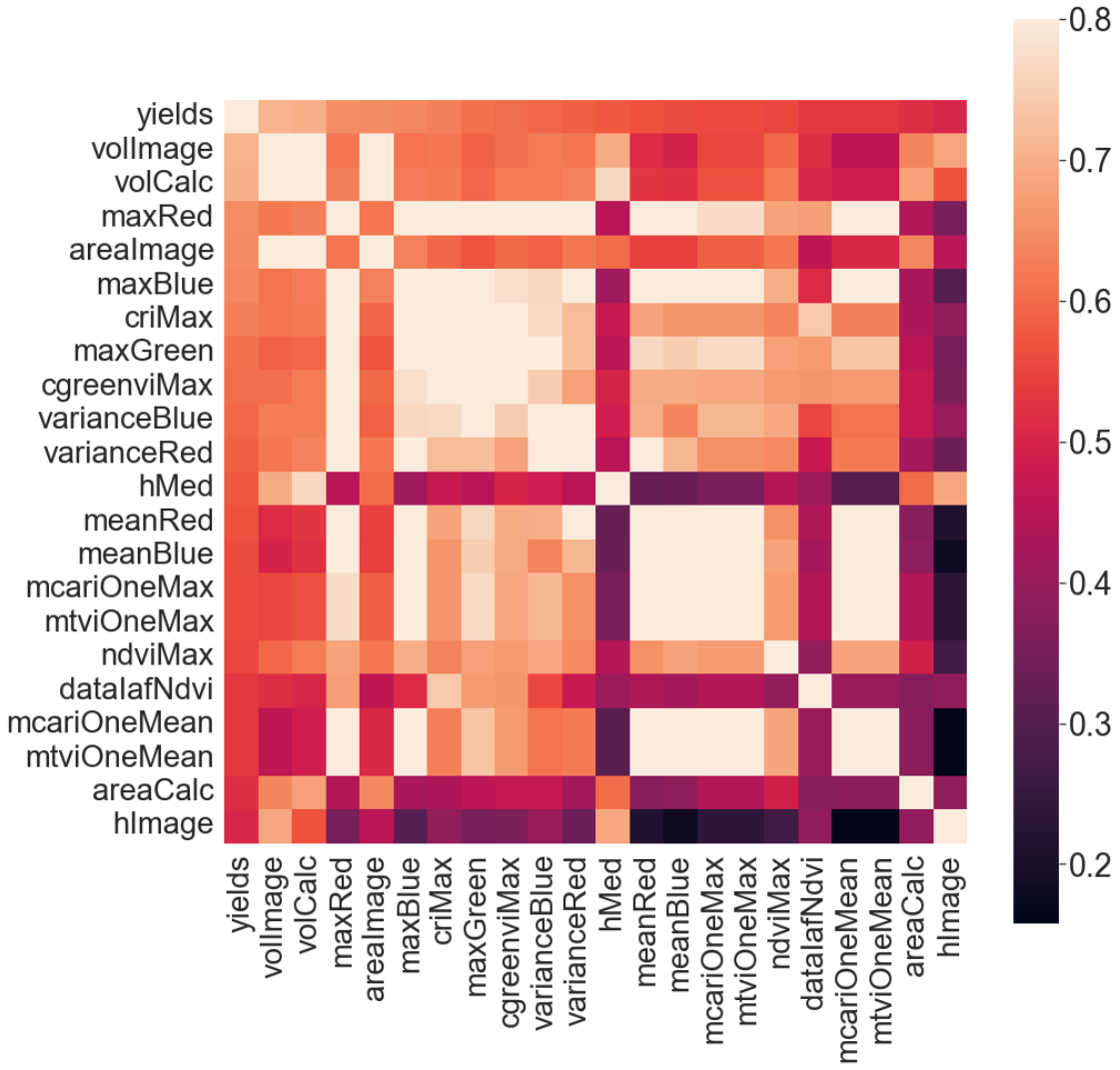


[ ]:
```
f, ax = pyplot.subplots(figsize=(20,20))
sns.heatmap(correlation,vmax=0.8,square=True)
```

[ ]: <AxesSubplot:>

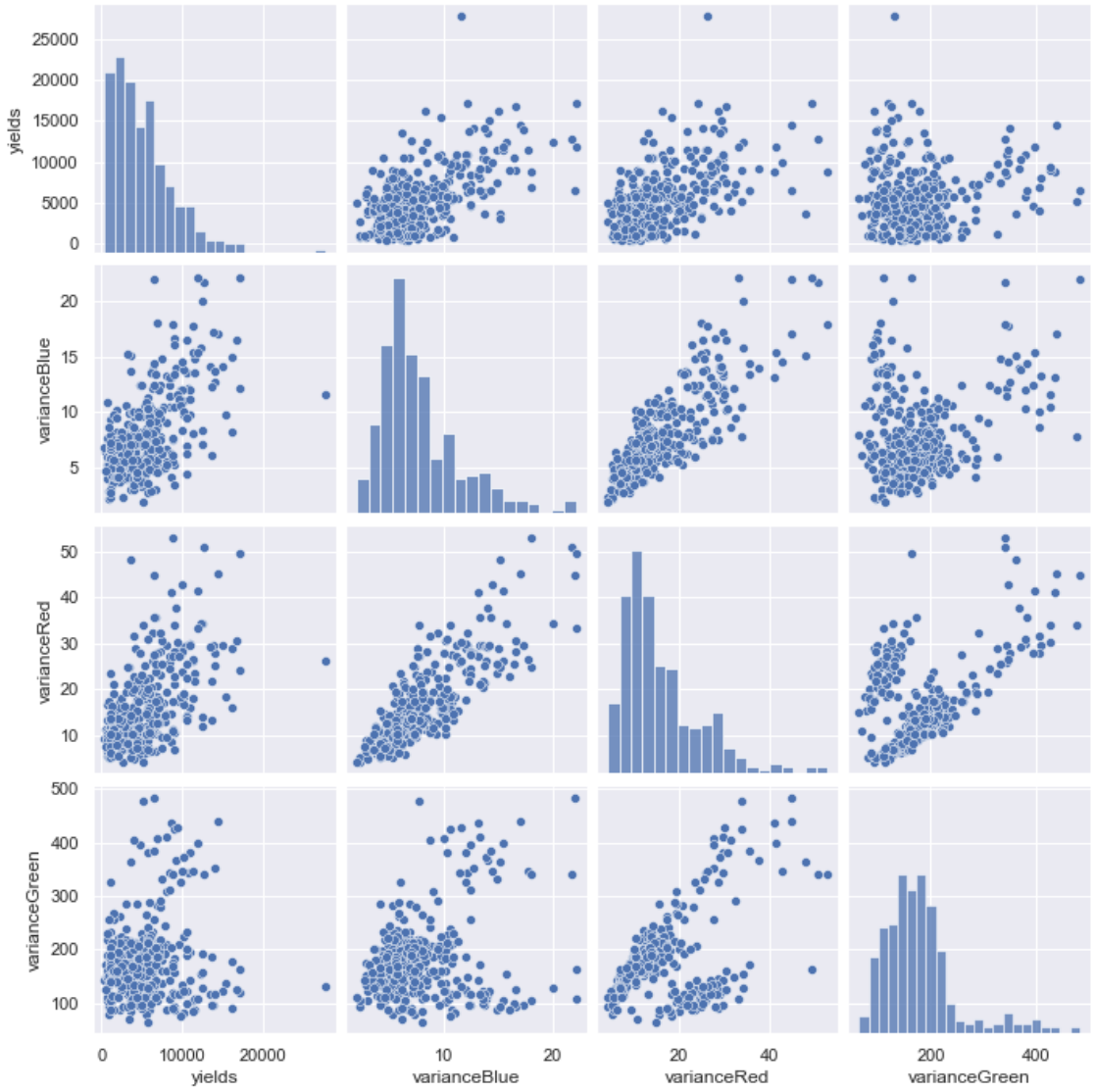


[ ]:
```
cols= correlation.nlargest(22,"yields")["yields"].index
cm= np.corrcoef(dataframe[cols].values.T)
f, ax = pyplot.subplots(figsize=(15,15))
sns.set(font_scale=2.6)
plot= sns.heatmap(cm,vmax=0.8,square=True,yticklabels=cols.
  ↪values,xticklabels=cols.values)
```



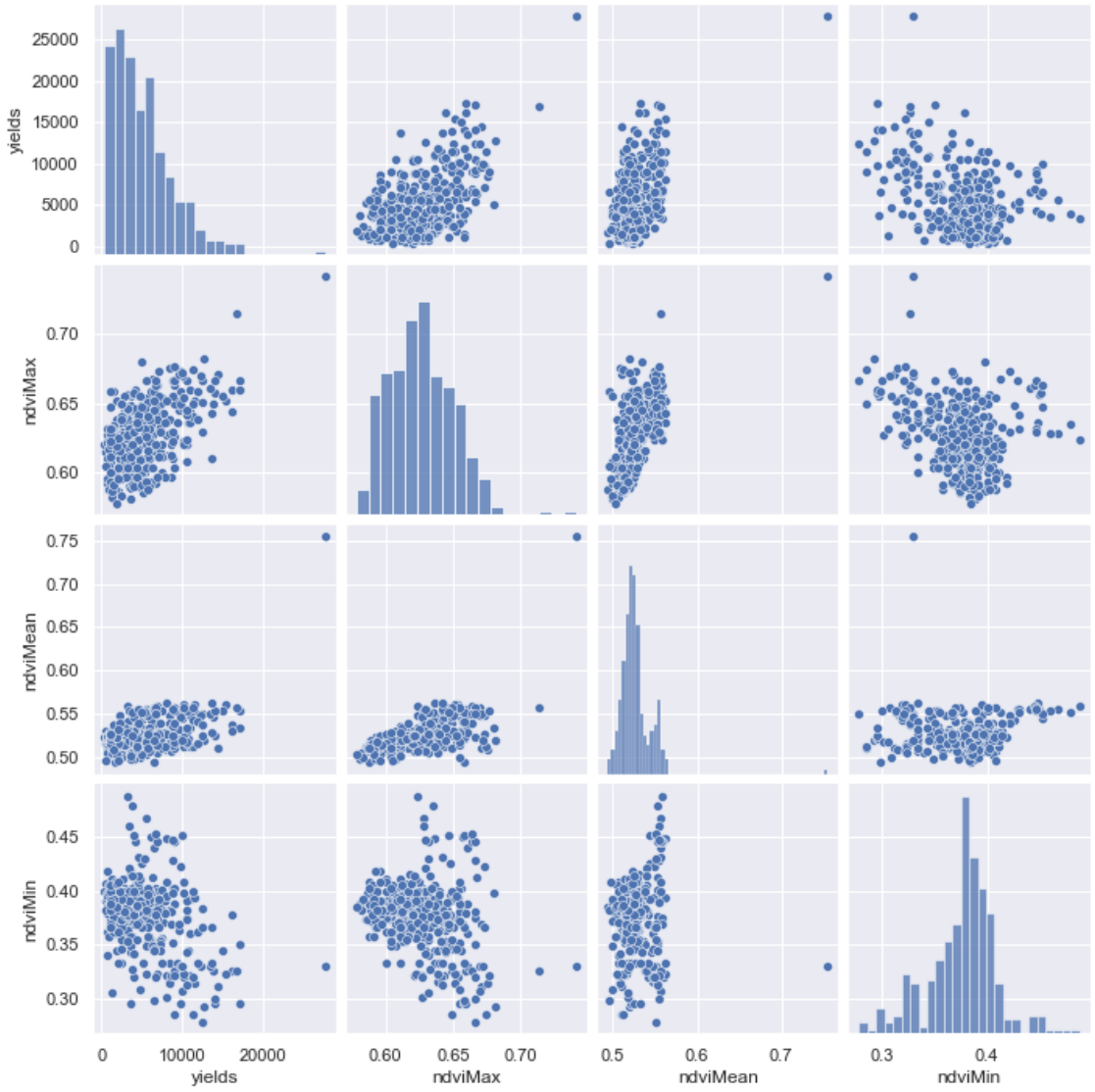
GRAFICAS DE VARIANZAS

[ ]:
```
sns.set()
colsVariance = ["yields","varianceBlue","varianceRed","varianceGreen"]
sns.pairplot(dataframe[colsVariance],height=2.5)
pyplot.show()
```



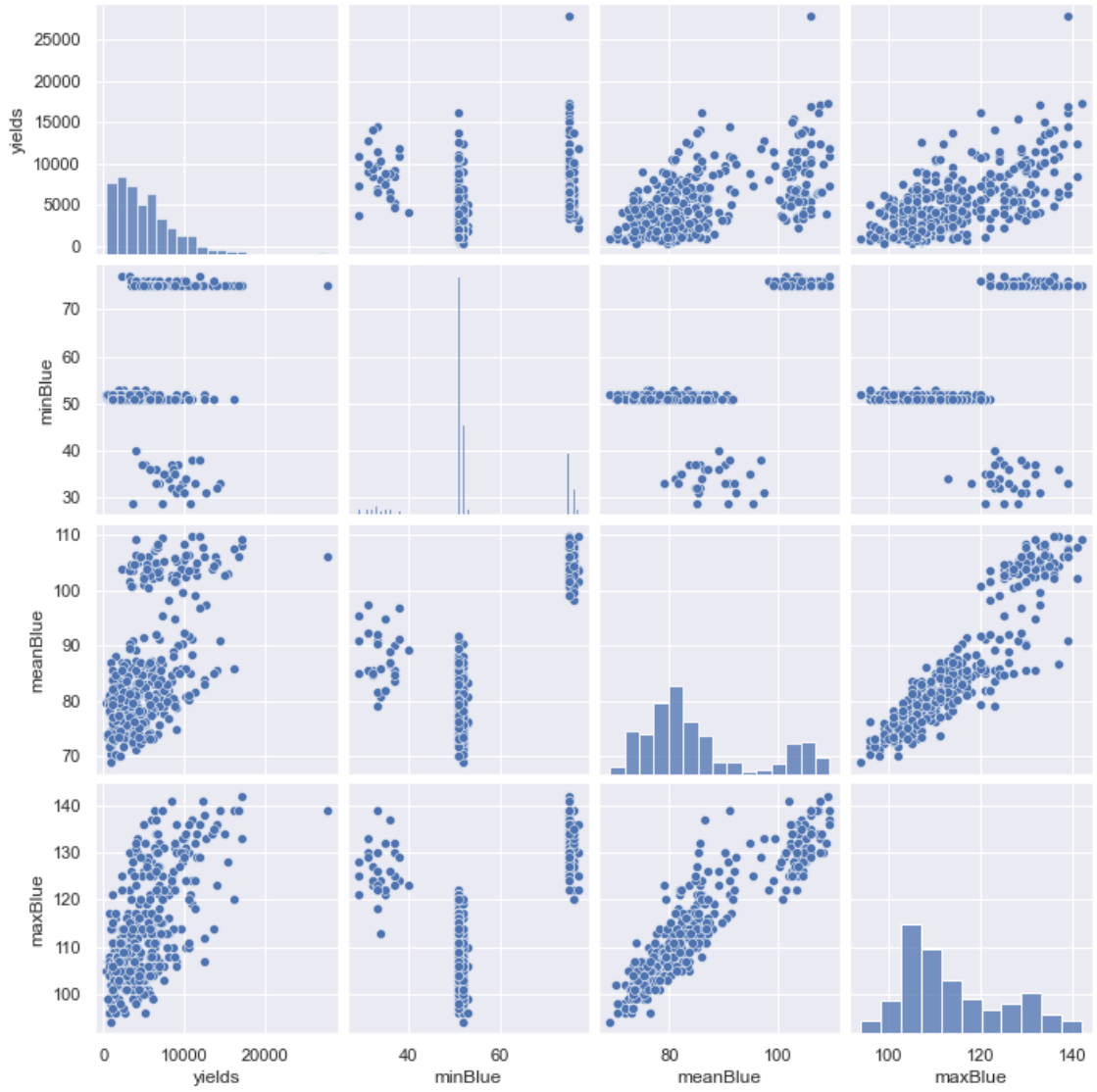
GRAFICA DE NDVI

[ ]:
```
sns.set()
colsNdvi = ["yields","ndviMax","ndviMean","ndviMin"]
sns.pairplot(dataframe[colsNdvi],height=2.5)
pyplot.show()
```

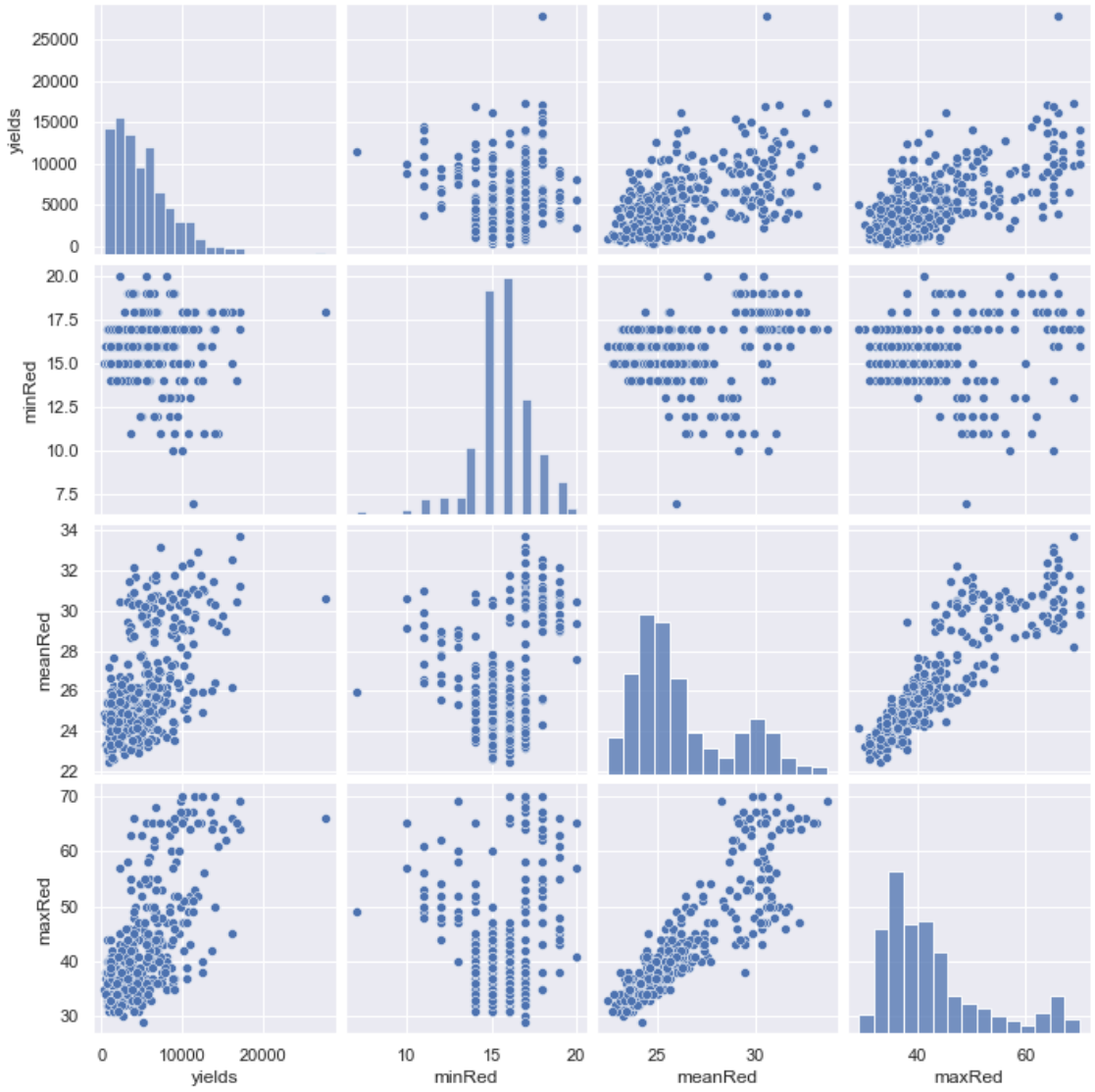


GRAFICA DE BANDAS

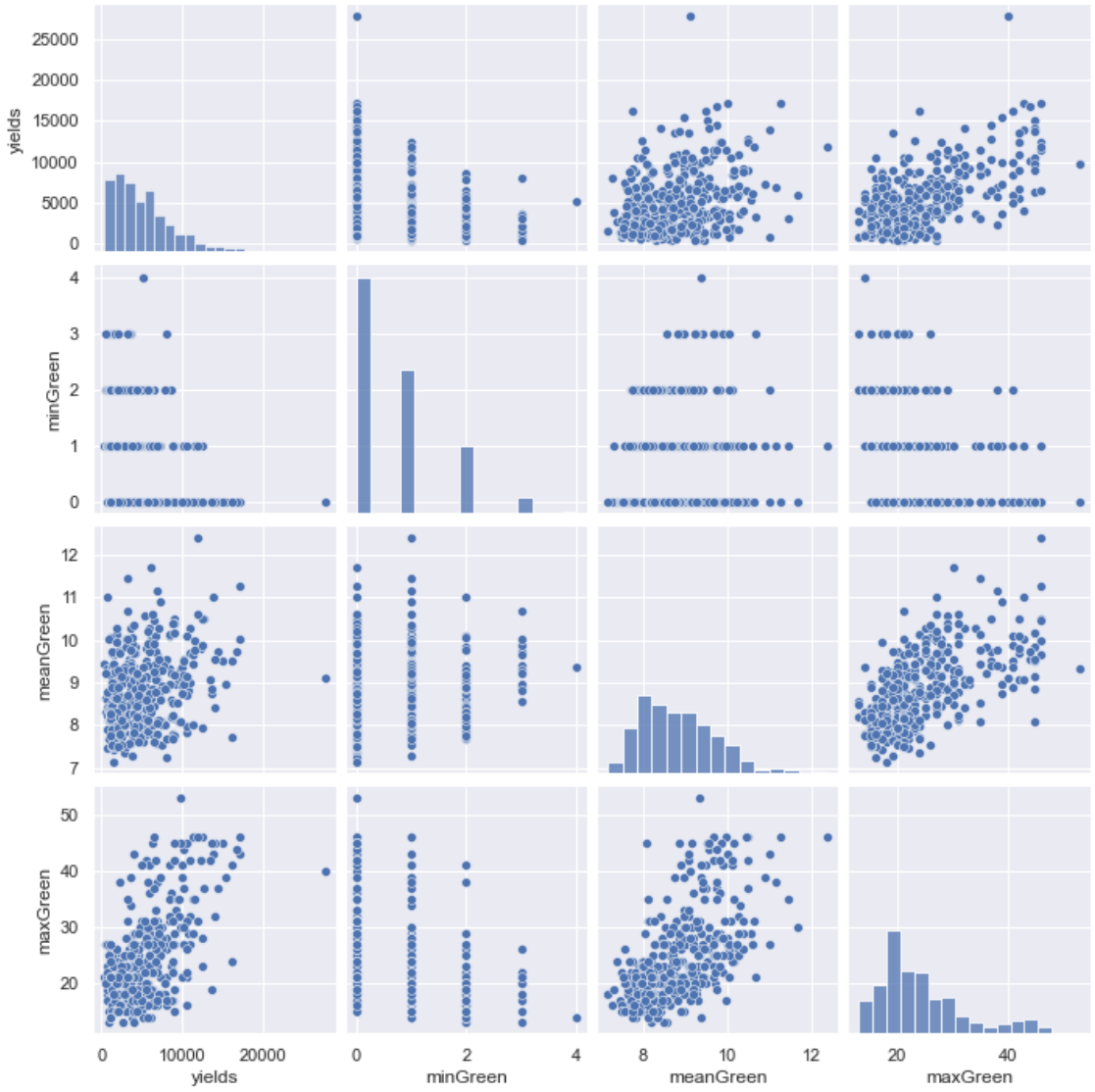
[ ]:
```
sns.set()
colsBlue = ["yields","minBlue","meanBlue","maxBlue"]
sns.pairplot(dataframe[colsBlue],height=2.5)
pyplot.show()
```



[ ]:
```
sns.set()
colsRed = ["yields","minRed","meanRed","maxRed"]
sns.pairplot(dataframe[colsRed],height=2.5)
pyplot.show()
```



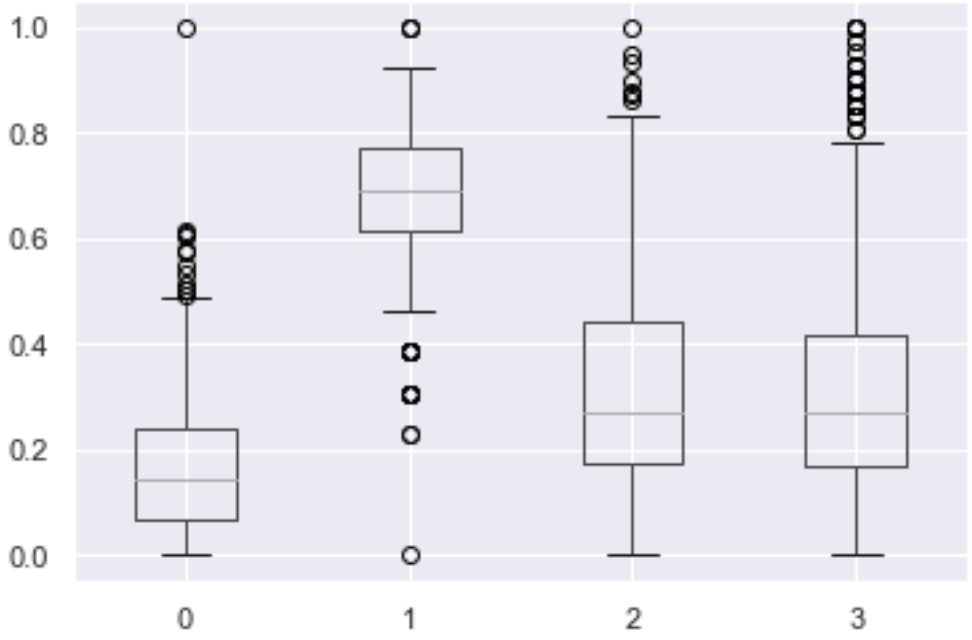
[ ]:
```
sns.set()
colsGreen = ["yields","minGreen","meanGreen","maxGreen"]
sns.pairplot(dataframe[colsGreen],height=2.5)
pyplot.show()
```



NORMALIZE

[ ]:
```
from sklearn.preprocessing import MinMaxScaler
x = dataframe[colsRed]
min_max_scaler = MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
df = pd.DataFrame(x_scaled)
df.boxplot()
```

[ ]: <AxesSubplot:>


1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3