DIPLOMATURA EN CIENCIA DE DATOS, APRENDIZAJE AUTOMÁTICO Y SUS APLICACIONES

MENTORÍA M03-2023

Descifrando el Universo: apariencia de las galaxias

Directora: Ingrid Vanessa Daza Perilla

Grupo 2: Ailín Asís, Joaquín Gamalerio y Pablo Velez.

INFORME:

Práctico Análisis Exploratorio y Curación de Datos

Se exploraron el conjunto de datos obtenidos a partir del **S**loan **D**igital **S**ky **S**urvey (https://skyserver.sdss.org/CasJobs/). El mismo contiene información de 92102 galaxias (*dataset*: "galaxias_1.csv")

Preparación de los datos

Para esta primera parte, se importaron las siguientes librerías*:*

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy as sp
import numpy as np
import missingno as msno
```

El *dataset* original consta de 92.102 registros y 14 variables y no posee valores faltantes.
Las variables son: 'objID', 'ra', 'dec', 'modelMag_u', 'modelMag_g', 'modelMag_r', 'modelMag_i', 'modelMag_z', 'petroR90_r', 'z', 'Color', 'elliptical', 'spiral' y 'uncertain'.

Se añade una columna con la clase de cada galaxia, asignando la etiqueta 'I' (de irregular) a las galaxias con etiqueta 'uncertain'

```
clase=[]
for i in range(0,data.shape[0]):
    if data.elliptical[i]==1:
        clase.append('E')
    elif data.spiral[i]==1:
        clase.append('S')
    elif data.uncertain[i]==1:
        clase.append('I')
data['clase']=clase
```

Se setea a la columna ‘objID’ como el id del dataframe y en un primer análisis descriptivo se observan algunos valores de objID repetidos:

```
data = data.set_index('objID')
data.drop(['elliptical','spiral','uncertain'],axis=1)
```

objID	ra	dec	modelMag_u	modelMag_g	modelMag_r	modelMag_i	modelMag_z	petroR90_r	z	Color	clase
1,23765119242489E+018	116.519097	39.886407	17.76235	16.72601	16.33972	16.06614	15.90478	8.393773	0.041521	-1.422625	S
1,23765149575578E+018	116.451900	41.421270	18.12179	16.26214	15.39272	14.97515	14.65105	9.674847	0.040211	-2.729061	I
1,23767370611537E+018	115.946713	41.918877	18.57293	17.42053	17.01788	16.75617	16.70899	11.277470	0.024386	-1.555044	I
1,2376737066523E+018	116.051943	42.287231	21.37438	19.77335	19.55791	20.35405	18.88184	1.539542	0.039137	-1.816479	I
1,23765127349266E+018	117.287392	43.434782	19.18845	17.99682	17.51119	17.26241	17.09056	12.471450	0.042591	-1.677259	I
...	...	...	...	...	...	...	...	...	...	...	...
1,23765494945271E+018	245.038742	50.552353	19.65135	17.74399	16.86763	16.43538	16.09720	3.514471	0.048618	-2.783718	I
1,23765153764514E+018	219.313588	62.338577	18.77944	17.71890	17.35237	17.14414	17.04369	4.985637	0.036884	-1.427076	I
1,237655744025E+018	239.043213	4.601942	18.93000	17.34099	16.79838	16.53403	16.52089	13.324690	0.034145	-2.131620	S
1,23765546806247E+018	229.410394	0.947489	19.52601	17.68973	16.90745	16.50974	16.21097	4.854905	0.039585	-2.618553	I
1,23765546806247E+018	229.410394	0.947489	19.52601	17.68973	16.90745	16.50974	16.21097	4.854905	0.039542	-2.618553	I

92102 rows × 11 columns

En concreto el id repetido es de 1,23765546806247E+018, todos los datos se repiten excepto los de la variable ‘z’. Esto puede deberse a que se adquirieron diferentes mediciones del mismo objeto, y estas pueden variar dependiendo de las condiciones climatológicas al momento de la medición.

A continuación se muestra el tipo de dato de cada columna:

```
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   objID       92102 non-null  object
 1   ra          92102 non-null  float64
 2   dec         92102 non-null  float64
 3   modelMag_u  92102 non-null  float64
 4   modelMag_g  92102 non-null  float64
 5   modelMag_r  92102 non-null  float64
 6   modelMag_i  92102 non-null  float64
 7   modelMag_z  92102 non-null  float64
 8   petroR90_r  92102 non-null  float64
 9   z           92102 non-null  float64
 10  Color       92102 non-null  float64
 11  elliptical  92102 non-null  int64
 12  spiral      92102 non-null  int64
 13  uncertain   92102 non-null  int64
 14  clase       92102 non-null  object
dtypes: float64(10), int64(3), object(2)
```

Análisis de valores duplicados:

Se encontraron 34421 objetos con id repetido. Debido a que nuestro dataset es lo suficientemente grande y para evitar conflictos entre datos, decidimos eliminar los duplicados y el dataset queda con 57681 datos, lo cual consideramos que, a nuestros fines, es una cantidad suficiente para realizar estadística.

Otras exploraciones del dataset.

Se observan nuevamente algunos valores atípicos o anormales, que serán tratados con más detalle a continuación.

	ra	dec	modelMag_u	modelMag_g	modelMag_r	modelMag_i	modelMag_z	petroR90_r	z	Color
count	57681.000000	57681.000000	57681.000000	57681.000000	57681.000000	57681.000000	57681.000000	57681.000000	57681.000000	57681.000000
mean	183.419958	25.380842	183.765182	172.718436	162.646477	170.157392	136.560776	54.057630	0.035991	-3.416101
std	57.659031	18.616160	1734.388143	1622.623065	1540.039554	1568.942667	1383.827414	882.310694	0.008498	72.031187
min	0.008745	-11.202394	-9999.000000	-9999.000000	11.902330	11.459980	-9999.000000	0.842248	0.020001	-2859.000000
25%	151.447181	10.476006	17.752040	16.314860	15.647160	15.287560	15.006410	6.132339	0.028957	-2.431772
50%	183.931359	23.898519	18.447040	17.119080	16.555560	16.247150	16.020130	8.386065	0.036099	-1.898378
75%	221.634978	39.408083	19.011310	17.738920	17.255280	16.983440	16.798670	11.335120	0.043573	-1.570280
max	359.965567	70.133213	25756.000000	20542.000000	19138.000000	23871.000000	20767.000000	78255.000000	0.050000	10015.860000

KNN sobre valores atípicos

se realizó el siguiente filtrado a los valores que deben someterse a imputación:

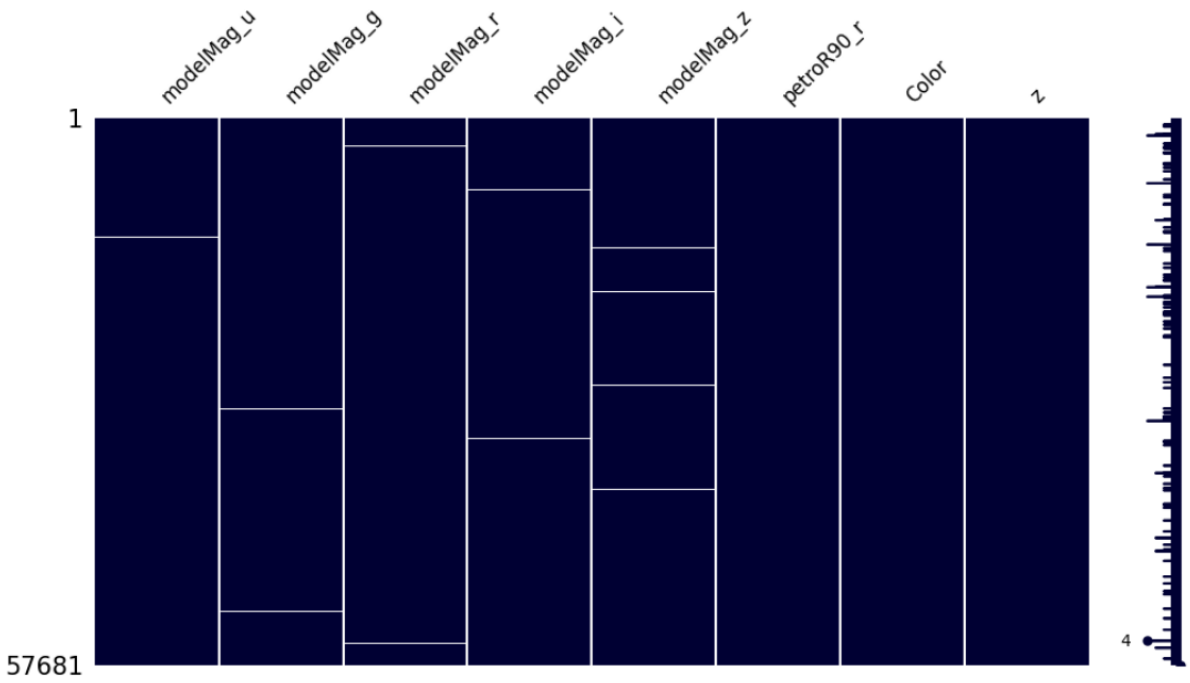
```
data.loc[(data.modelMag_u <0), 'modelMag_u'] = np.nan
data.loc[(data.modelMag_g <0), 'modelMag_g'] = np.nan
data.loc[(data.modelMag_r <0), 'modelMag_r'] = np.nan
data.loc[(data.modelMag_i <0), 'modelMag_i'] = np.nan
data.loc[(data.Color <-100), 'Color'] = np.nan
data.loc[(data.modelMag_z <0), 'modelMag_z'] = np.nan
```

```
data.loc[(data.modelMag_u >22), 'modelMag_u'] = np.nan
data.loc[(data.modelMag_g >22.0), 'modelMag_g'] = np.nan
data.loc[(data.modelMag_r >22.2), 'modelMag_r'] = np.nan
data.loc[(data.modelMag_i >21.3), 'modelMag_i'] = np.nan
data.loc[(data.Color >20), 'Color'] = np.nan
data.loc[(data.modelMag_z >20.5), 'modelMag_z'] = np.nan
data.loc[(data.petroR90_r <0), 'petroR90_r'] = np.nan
```

Como se observa, se estableció la cota inferior de 0 a las magnitudes aparentes y al tamaño, tambien se filtraron valores atípicos de color. Las cotas superiores aplicadas a las magnitudes aparentes se justifican en la siguiente imagen, extraída de https://classic.sdss.org/dr4/ .

Average wavelengths and magnitude limits (95% detection repeatability for point sources)	*u*	*g*	*r*	*i*	*z*
	3551Å	4686Å	6165Å	7481Å	8931Å
	22.0	22.2	22.2	21.3	20.5

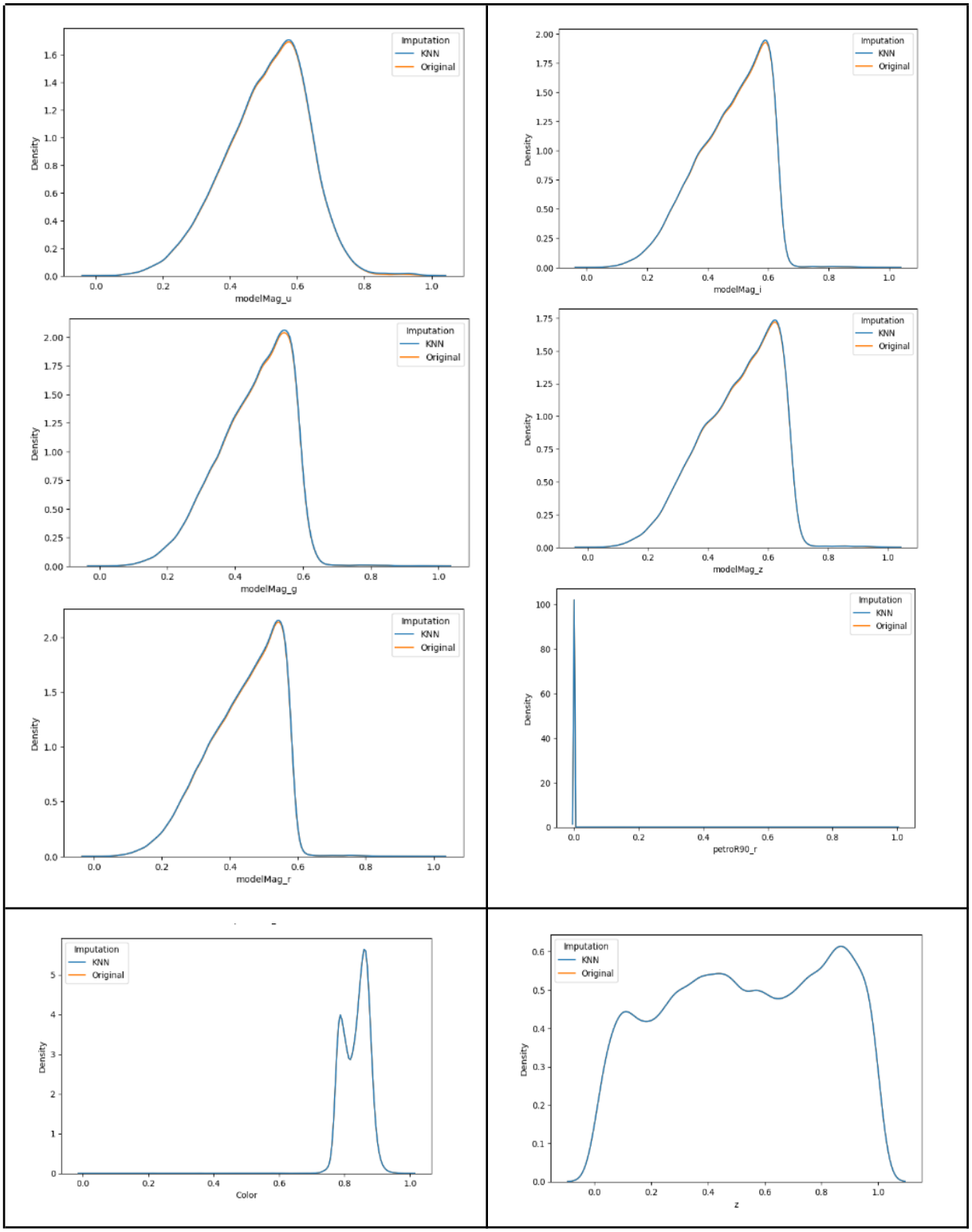
Utilizando la librería missingno para poder visualizar de mejor manera los gráficos a imputar se muestra a continuación la distribución de dichos datos.



Se importan las librerías necesarias para la imputación

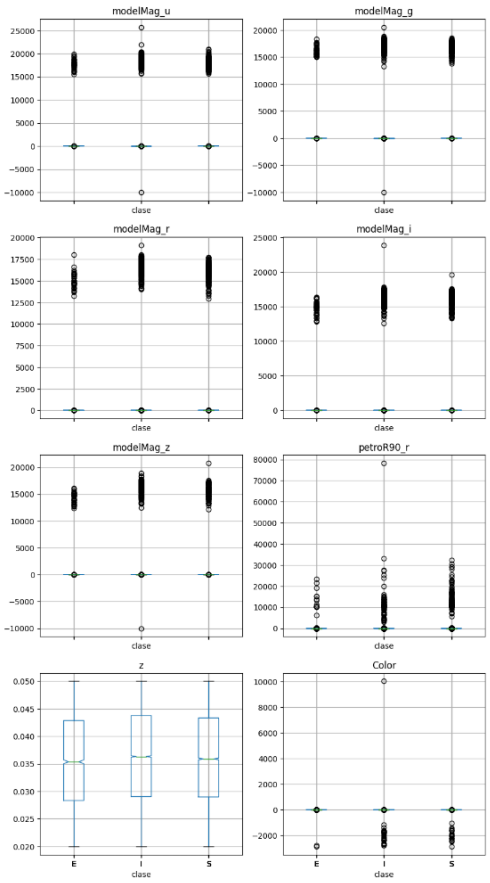
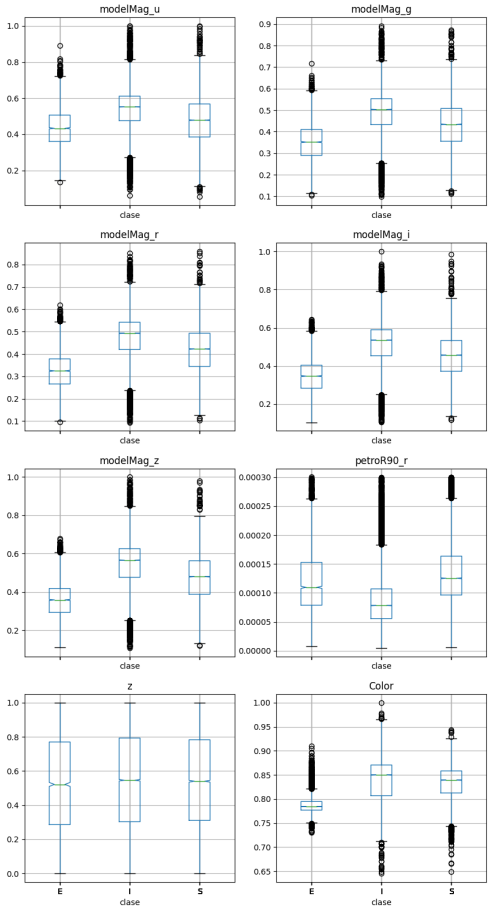
```
from sklearn.preprocessing import MinMaxScaler
from sklearn.experimental import enable_iterative_imputer
from sklearn.neighbors import KNeighborsRegressor
from sklearn.impute import IterativeImputer
```

Se escalan los datos y se procede a realizar la imputación utilizando todas las columnas y se obtuvo el siguiente gráfico de comparación de densidades.



A continuación se eliminaron los outliers con el criterio $Q1*2.5 < x < Q3*2.5$ y se creó el dataframe df1, de 56545 datos, es decir, se eliminaron el 1.97% de datos.

El resultado inicial y final se muestra en la siguiente imagen.

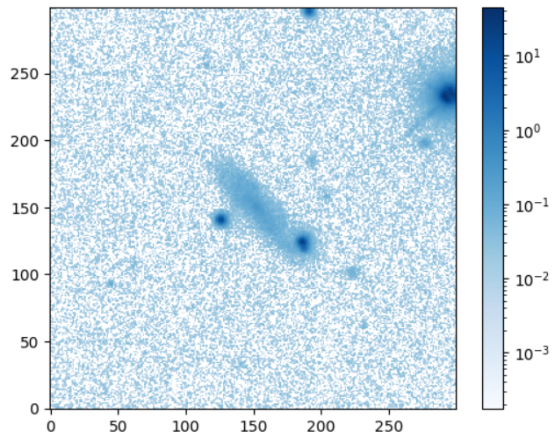
Original	*Final*
	

Visualización de Imágenes

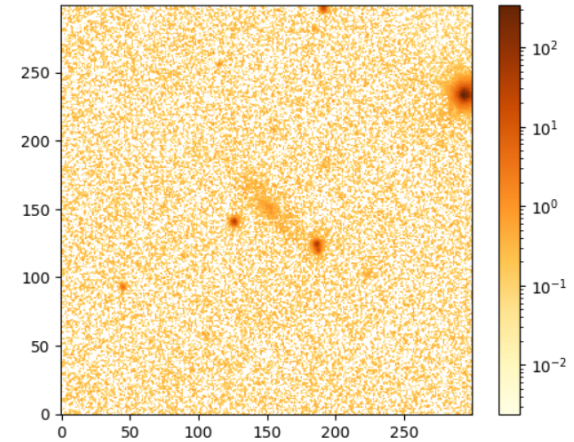
Las imágenes tienen un tamaño de 300x300 y poseen 5 filtros 'SDSSg', 'SDSSi', 'SDSSr', 'SDSSu', 'SDSSz'.

Para poder trabajar con estas imágenes se necesitan los siguientes paquetes

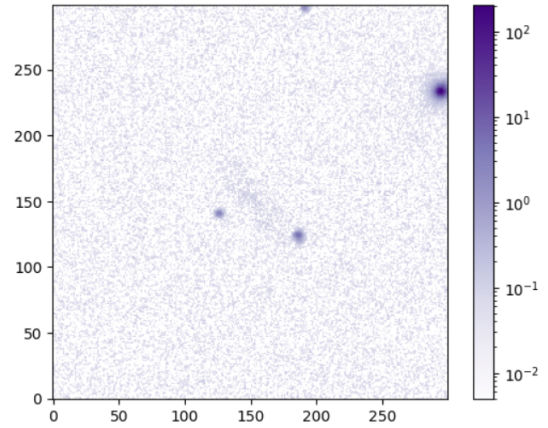
```
from astropy.coordinates import SkyCoord
from matplotlib.colors import LogNorm
import pandas as pd
import matplotlib.pyplot as plt
from astropy.io import ascii
from astropy import units
from astroquery.skyview import SkyView
import numpy as np
from astropy import units as u
```



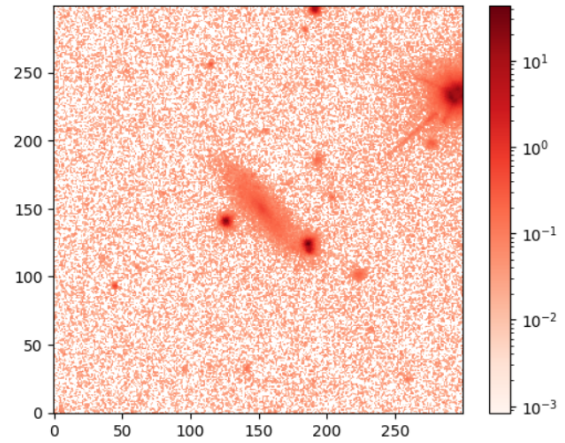
SDSSg



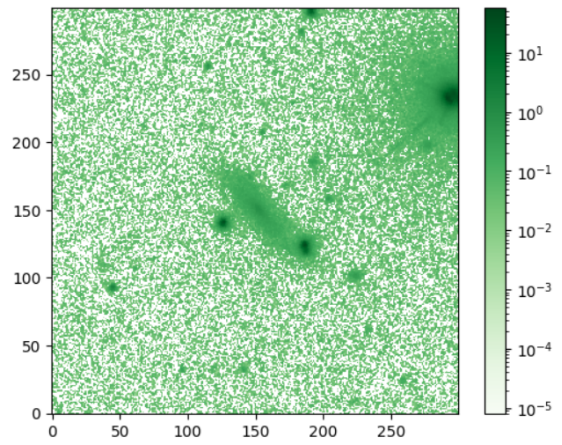
SDSSz



SDSSu



SDSSr



SDSSi

Se puede visualizar cada imagen en su correspondiente filtro.

Se genera un dataframe (df2) en donde cada fila es una imagen y las columnas son los valores de cada px por filtro, por lo tanto, si las imágenes tienen un tamaño (Xpx, Ypx) y cuenta con f filtros, entonces la cantidad de columnas de df2 son Xpx*Ypx*f:

```
def image_transformation(ra, dec):
    coords_gx = SkyCoord(ra, dec, unit=(u.deg, u.deg))
    img = SkyView.get_images(position=coords_gx, survey=['SDSSg', 'SDSSi', 'SDSSr', 'SDSSu', 'SDSSz'])
    SDSSg = img[0][0].data.flatten()
    SDSSi = img[0][0].data.flatten()
    SDSSr = img[0][0].data.flatten()
    SDSSu = img[0][0].data.flatten()
    SDSSz = img[0][0].data.flatten()

    result = np.concatenate([SDSSg,SDSSi,SDSSr,SDSSu,SDSSz])

    return result
```

Se descargaron solo 20 imágenes para demostrar el procedimiento, ya que resulta imposible descargar todas con nuestros recursos limitados.

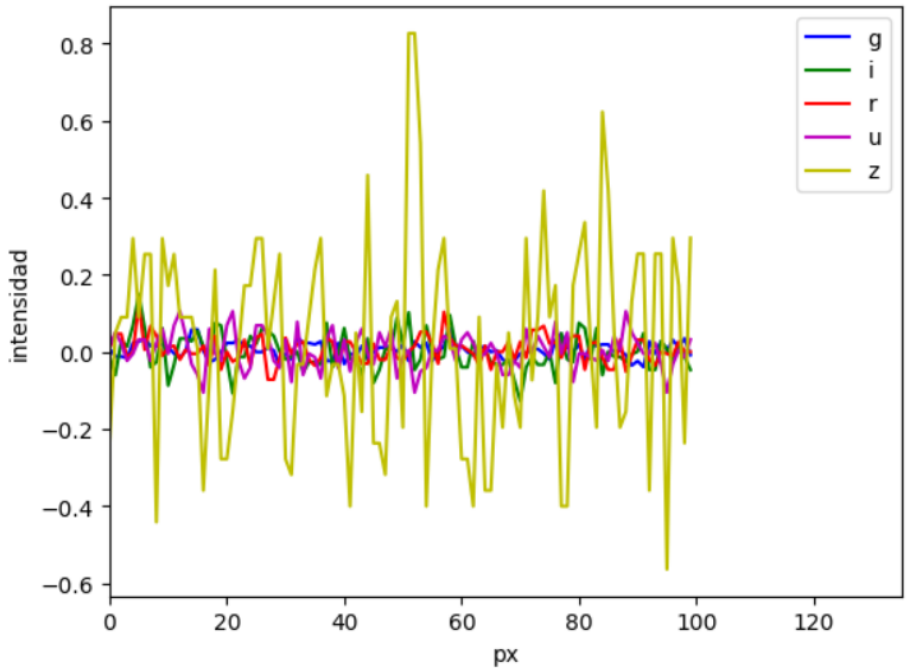
```
for index in range(1,20):
   # print(index)
    img = image_transformation(df1.ra.iloc[index], df1.dec.iloc[index])
    stack=np.vstack([stack,img])
    df2 =pd.DataFrame(stack)
```

```
df2.head(5)
```

	0	1	2	3	4	5	6	7	8	9	...	449990	449991
0	0.008057	-0.011337	-0.011337	-0.015228	0.023560	0.027466	0.035217	0.015808	0.015808	0.019684	...	-0.024597	0.001856
1	-0.011612	0.017517	-0.007965	-0.004326	0.024811	0.006607	0.002964	0.002964	-0.018890	-0.022522	...	0.017792	0.014145
2	-0.010437	-0.010437	0.008438	0.008438	0.000881	0.008423	-0.025543	0.004654	-0.006676	0.012207	...	0.000863	-0.010468
3	0.014923	0.007248	-0.004265	0.034119	-0.004265	0.011078	0.026428	0.030243	0.076294	-0.000431	...	0.176270	0.180176
4	0.005363	-0.009720	0.009140	0.009140	-0.005951	0.012909	0.001591	-0.028564	-0.002182	0.009125	...	-0.020905	-0.051086

5 rows × 450000 columns

Utilizando df2 se puede tomar una sola fila y mostrar a partir de un gráfico la distribución de valores de los px en cada filtro en un rango de pxs:



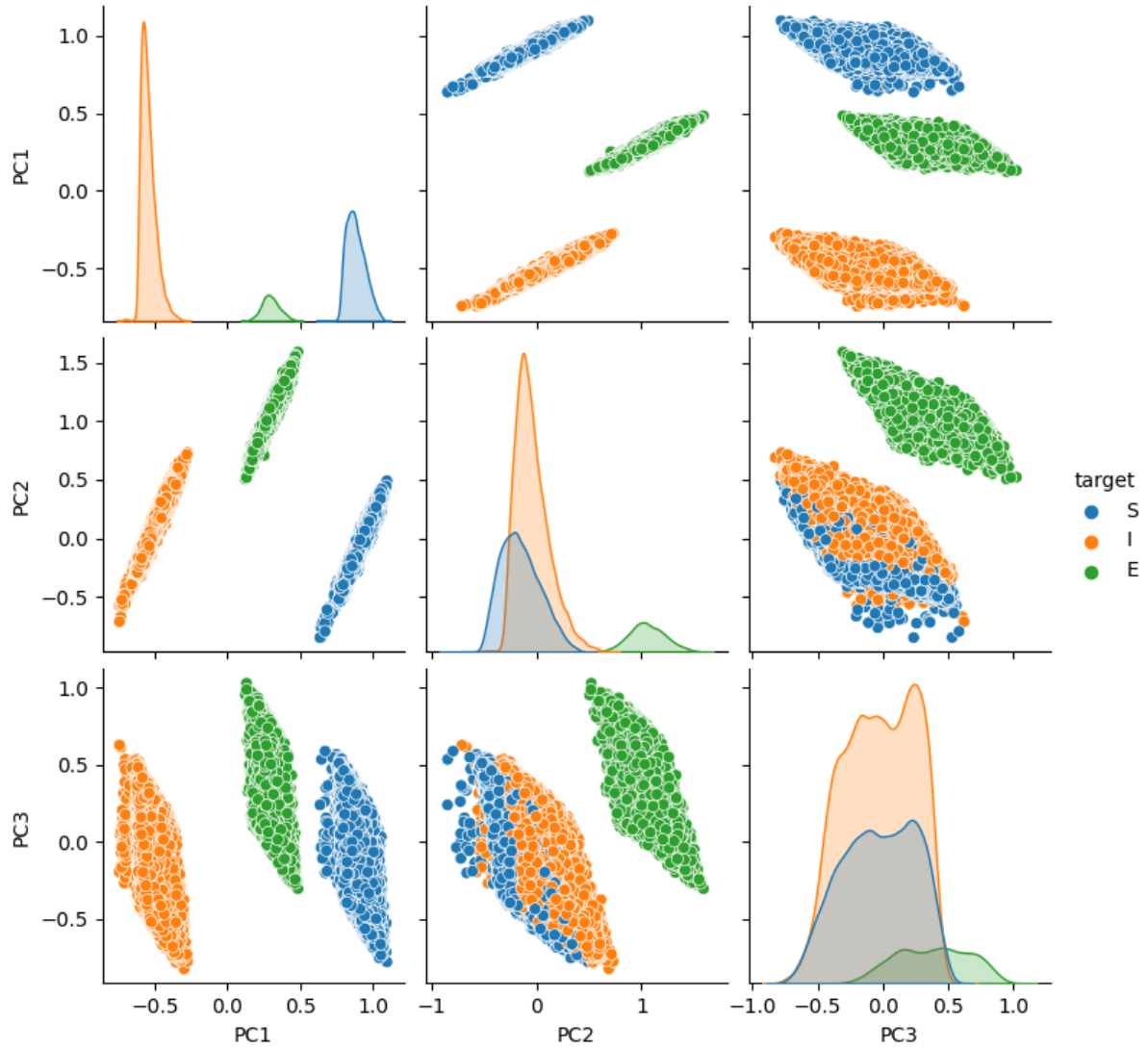
Por último se une df1 con df2 y se guarda como csv:

- result = pd.merge(df1, df2, left_index=True, right_index=True)
- result.to_csv('result.csv

PCA

Se realizó PCA con 3 componentes principales, sobre el dataframe df1, específicamente sobre las caracterisicas 'modelMag_u', 'modelMag_g', 'modelMag_r', 'modelMag_i', 'modelMag_z', 'petroR90_r', 'z','Color','ra','dec','elliptical','spiral','uncertain'.

El resultado es el siguiente:



Se observa una clara separación de clases lo cual sugiere que el problema puede ser encarado reduciendo la dimensionalidad a 3 componentes principales y así ahorrar poder de cómputo.

Conclusiones

Se estudió el *dataset*: “galaxias_1.csv”, realizando una tarea de visualización y limpieza, empezando por la eliminación de los datos repetidos, continuando por la imputación de valores anormales y finalizando por la limpieza de los outliers.

Luego se estudió la visualización de imágenes de galaxias y la distribución de intensidad de px en cada filtro.

Por último se realizó un análisis de componentes principales al dataset, encontrando que 3 componentes son suficientes para caracterizar al conjunto de datos.