Current Data Analysis Methods, MSc program, 2018

Homework 1-5. Sloan Digital Sky Survey RD14 Data Analysis.

Dementyeva Slavyana

---

1. Homework 1

1.1 Explanation of the dataset

Table 1 Dataset profile

Data source:	The Sloan Digital Sky Survey
Source description	The Sloan Digital Sky Survey has created the most detailed three dimensional maps of the Universe ever made, with deep multi-color images of one third of the sky, and spectra for more than three million astronomical objects
Link	https://www.sdss.org/dr14/
Licence	Creative Commons Attribution license (CC-BY)
Dataset description:	1000 entities 17 features + 1 class columns (can be used to check the classification results)

Object description
Each object is either a star, galaxy or quasar.
Feature description:
(https://www.kaggle.com/lucidlenn/sloan-digital-sky-survey/home)
- objid = Object Identifier
- ra = J2000 Right Ascension (r-band)
- dec = J2000 Declination (r-band)

Right ascension (abbreviated RA) is the angular distance measured eastward along the celestial equator from the Sun at the March equinox to the hour circle of the point above the earth in question. When paired with declination (abbreviated dec), these astronomical coordinates specify the direction of a point on the celestial sphere (traditionally called in English the skies or the sky) in the equatorial coordinate system.
Source: https://en.wikipedia.org/wiki/Right_ascension
- u = better of DeV/Exp magnitude fit
- g = better of DeV/Exp magnitude fit
- r = better of DeV/Exp magnitude fit
- i = better of DeV/Exp magnitude fit
- z = better of DeV/Exp magnitude fit

The Thuan-Gunn astronomic magnitude system. u, g, r, i, z represent the response of the 5 bands of the telescope.
Further education: https://www.astro.umd.edu/~ssm/ASTR620/mags.html
- run = Run Number
- rereun = Rerun Number

• camcol = Camera column
• field = Field number
Run, rerun, camcol and field are features which describe a field within an image taken by the SDSS. A field is basically a part of the entire image corresponding to 2048 by 1489 pixels. A field can be identified by:
- run number, which identifies the specific scan, - the camera column, or "camcol," a number from 1 to 6, identifying the scanline within the run, and - the field number. The field number typically starts at 11 (after an initial rampup time), and can be as large as 800 for particularly long runs. - An additional number, rerun, specifies how the image was processed.
View "SpecObj"
• specobjid = Object Identifier
• class = object class (galaxy, star or quasar object)
The class identifies an object to be either a galaxy, star or quasar. This will be the response variable which we will be trying to predict.
• redshift = Final Redshift
• plate = plate number
• mjd = MJD of observation
• fiberid = fiber ID
In physics,redshift happens when light or other electromagnetic radiation from an object is increased in wavelength, or shifted to the red end of the spectrum.
Featured problems
The provided set can be used to distinguish between given objects: based on the data, it can be classified as a star, galaxy or quasar

2. Homework 2

2.1 Applying k-means
The Python code on this part is provided in **AppendixA.**
Choosing features
List of features chosen for the k-means algorithm (6 out of 16):
['u', 'g', 'r', 'i', 'z', 'redshift']
The features chosen represent the magnitude of the sky object which is the measure of the brightness. This measure can be used to distinguish between sky objects of different clusters (e.g. Stars, Galaxies or Quasars).
Data standardization
To balance contributions of the features the **mean/range standardization** was applied.
The characteristics used in standardization algorithm are provided in the Table 1.

Table 2. Characteristics of the Dataset

Characteristics	u	g	r	i	z	redshift
Mean	18.6194	17.3719	16.841	16.5836	16.4228	0.143726
Max	19.5999	19.919	24.802	28.1796	22.8331	5.35385
Min	12.989	12.7995	12.4316	11.9472	11.6104	-0.00413608
Midrange	16.2944	16.3593	18.6168	20.0634	17.2217	2.67486
std	0.828656	0.945457	1.06776	1.1418	1.20319	0.388774
range	6.61093	7.11942	12.3704	16.2324	11.2226	5.35799

Percentage of objects fallen into each of the group
Number of clusters K = 5

Cluster	# of objects	% of objects
1	2144	21.44
2	2898	28.98
3	4958	49.58

Number of clusters K = 3

Cluster	# of objects	% of objects
1	937	9.37
2	2908	29.08
3	3250	32.50
4	736	7.36
5	2169	21.69

2.2 Interpreting partitions

There are no nominal features in the dataset, thus only the relative difference is calculated and is used for the interpretation.
The data on relative distances for different partitions and features is provided below.
Our focus will be on feature 'redshift' for the case of 3 clusters.

Number of clusters K = 5

Cluster № 1

	u	g	r	i	z	redshift
ckv	16.8654	15.4638	14.9278	14.7145	14.5574	0.0148811
cv	18.6194	17.3719	16.841	16.5836	16.4228	0.143726
difference	-1.75399	-1.90817	-1.91316	-1.86911	-1.86548	-0.128845
dkv	-9.42026	-10.9842	-11.3601	-11.2709	-11.3591	-89.6462

Cluster № 2

	u	g	r	i	z	redshift
ckv	18.9601	17.4325	16.7205	16.3731	16.1373	0.0640329
cv	18.6194	17.3719	16.841	16.5836	16.4228	0.143726
difference	0.34074	0.0605319	-0.120426	-0.210464	-0.285548	-0.0796928
dkv	1.83003	0.348447	-0.715076	-1.26911	-1.73873	-55.4478

Cluster № 3

	u	g	r	i	z	redshift
ckv	19.0974	18.8971	18.8164	18.7405	18.7108	1.29946
cv	18.6194	17.3719	16.841	16.5836	16.4228	0.143726
difference	0.478064	1.52515	1.97543	2.15688	2.28799	1.15573
dkv	2.56757	8.77937	11.7299	13.0061	13.9318	804.122

Cluster № 4

	u	g	r	i	z	redshift
ckv	19.0974	18.8971	18.8164	18.7405	18.7108	1.29946
cv	18.6194	17.3719	16.841	16.5836	16.4228	0.143726
difference	0.478064	1.52515	1.97543	2.15688	2.28799	1.15573
dkv	2.56757	8.77937	11.7299	13.0061	13.9318	804.122

Cluster № 5

	u	g	r	i	z	redshift
ckv	17.983	16.6071	16.0562	15.8035	15.6485	0.0364029
cv	18.6194	17.3719	16.841	16.5836	16.4228	0.143726
difference	-0.636398	-0.764831	-0.784766	-0.780108	-0.774359	-0.107323
dkv	-3.41794	-4.40268	-4.65987	-4.7041	-4.71514	-74.6719

Number of clusters K = 3

In the second case the number of clusters Kwas was chosen to be equal 3, not 9, due to the known number of clusters in the original dataset (the source data contains “Class” column with either “Galaxy”, “Star” or “SQO” (quasar) value assigned to the each observation of a space object).

Cluster № 1

Redshift is considered as **Vk-** and is **much smaller** for this cluster than for the others.

	u	g	r	i	z	redshift
ckv	17.389	15.988	15.4406	15.2053	15.0484	0.0250913
cv	18.6194	17.3719	16.841	16.5836	16.4228	0.143726
difference	-1.23037	-1.3839	-1.40037	-1.37828	-1.37447	-0.118634
dkv	-6.60801	-7.9663	-8.31527	-8.31114	-8.36926	-82.5422

Cluster № 2

Redshift is considered as **Vk+** and is **much greater** for this cluster than for the others.

	u	g	r	i	z	redshift
ckv	19.1756	18.3741	18.0588	17.889	17.8056	0.376629
cv	18.6194	17.3719	16.841	16.5836	16.4228	0.143726
difference	0.556281	1.00221	1.21786	1.30545	1.3828	0.232904
dkv	2.98765	5.76911	7.23154	7.87196	8.41997	162.047

Cluster № 3

	u	g	r	i	z	redshift
ckv	18.8263	17.3846	16.7347	16.4165	16.2089	0.0588926
cv	18.6194	17.3719	16.841	16.5836	16.4228	0.143726
difference	0.2069	0.0126442	-0.106285	-0.167035	-0.213894	-0.0848331
dkv	1.11121	0.0727854	-0.631113	-1.00723	-1.30242	-59.0243

2.3 Feature comparison using Bootstrap

-

3. Homework 3

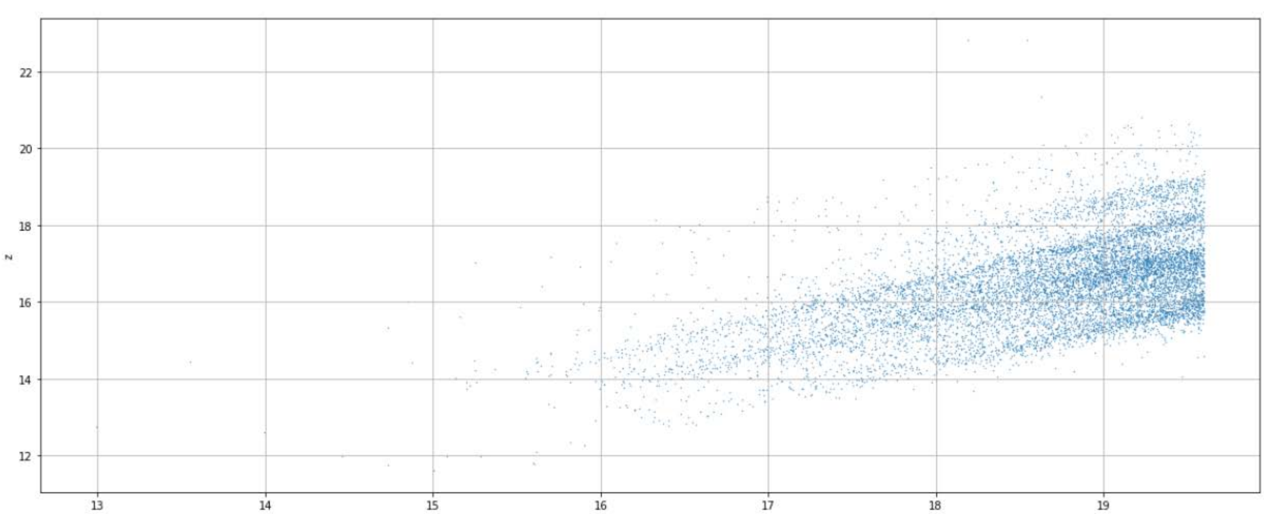
-

4. Homework 4

-

5. Homework 5

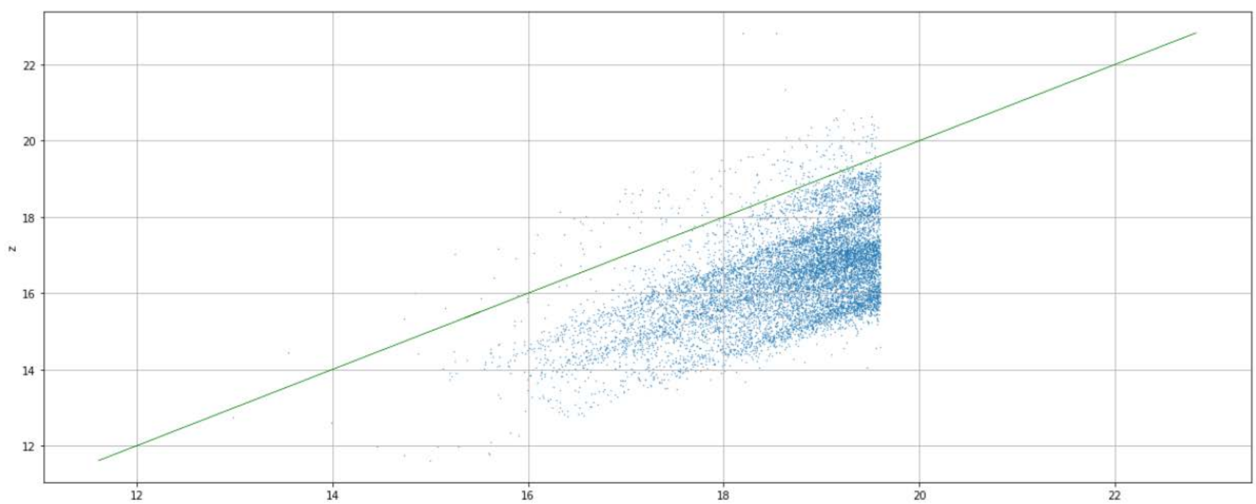
5.1 Scatter plot for feature ‘u’ and ‘z’ (components of magnitude)



5.1 Linear regression

Code example for linear regression is provided in Appendix C.

The slope of the line is close to 45 degrees, meaning that the relation between features is linear.



Appendix A. K-means code example.

```
In [ ]: import pandas as pd
        import numpy as np
        import random
        from scipy.spatial import distance
        from sklearn.cluster import KMeans
```

```
In [ ]: data = pd.read_csv('Sky.csv')
```

```
In [ ]: for col in data.columns:
            data[col] = pd.to_numeric(data[col], errors = 'ignore')
```

```
In [ ]: for col in data.columns:
            nun = data[col].nunique(dropna = True)
            print(col, nun, sep = "|")
```

```
In [ ]: data.head()
```

Choosing the features

```
In [ ]: features = ['u', 'g', 'r', 'i', 'z', 'redshift']
```

```
In [ ]: dataclast = data[features].copy()
```

Standardization

```
In [ ]: #Calculating parameters for the standardization
        standpar = pd.DataFrame(columns = dataclast.columns[1:len(dataclast.columns)], index = ['Mean', 'Max', 'Min', 'Midrange', 'std',
         'range'])
```

```
In [ ]: for i in standpar.columns:
            standpar[i]['Mean'] = dataclast[i].mean()
            standpar[i]['Max'] = dataclast[i].max()
            standpar[i]['Min'] = dataclast[i].min()
            standpar[i]['Midrange'] =  (standpar[i]['Max'] + standpar[i]['Min'])/2
            standpar[i]['std'] =  dataclast[i].std()
            standpar[i]['range'] =  standpar[i]['Max'] - standpar[i]['Min']
```

Range-related standardization

```
In [ ]: dataclast_rls = dataclast.copy()
        for i in standpar.columns:
            for row in dataclast_rls.index:
                dataclast_rls[i][row] = (dataclast_rls[i][row] - standpar[i]['Midrange'])/standpar[i]['range']
```

```
In [ ]: dataclast_rls.head()
```

Mean/Range standardization

In [ ]:
```
dataclast_mrs = dataclast.copy()
for i in standpar.columns:
    for row in dataclast_mrs.index:
        dataclast_mrs[i][row] = (dataclast_mrs[i][row] - standpar[i]['Mean'])/standpar[i]['range']
```

In [ ]:
```
dataclast_mrs.head()
```

Z-scoring standardization

In [ ]:
```
dataclast_zss = dataclast.copy()
for i in standpar.columns:
    for row in dataclast_zss.index:
        dataclast_zss[i][row] = (dataclast_zss[i][row] - standpar[i]['Mean'])/standpar[i]['std']
```

In [ ]:
```
dataclast_zss.head()
```

K-means with random initializations

In [ ]:
```
# list of chosen features
features = standpar.columns.values
```

Randomly assigning centroids

In [ ]:
```
clusnum = 5 # number of clusters
centroids = pd.DataFrame(index = range(1,6), columns = features)
for i in range(1,clusnum+1):
    c = random.randint(0, len(dataclast_rls))
    d = dataclast_rls[features]
    centroids.loc[i] = d.loc[c]
```

In [ ]:
```
k.loc[0]
```

In [ ]:
```
# euclidian distance
def eudist(x,y, feat):
    dist = 0
    for f in feat:
        dist+=(((x[f] - y[f]))**2)
    dist = math.sqrt(dist)
    return(dist)
```

In [ ]:
```
dataclast_rls['clust'] = np.nan # column with cluster number
```

In [ ]:
```
centroidsOld = centroids
centroidsnew = pd.DataFrame(data = np.zeros(shape = (clusnum, len(features))), index = range(1,6), columns = features)
# Loop for convergence over the k-means criterion
maxiter = 2
iteration = 0
dataclast_rls_init = pd.DataFrame(data = np.zeros(shape = (dataclast_rls[features].shape)), index = dataclast_rls[features].inde
x.values, columns = dataclast_rls[features].columns.values)
while iteration <= maxiter or (dataclast_rls_init != dataclast_rls).all:
        dataclast_rls_init = dataclast_rls
        for r in dataclast_rls.index:
            classdist = []
            for c in centroidsOld.index.values:
                dist = distance.euclidean(dataclast_rls[features].loc[r], centroidsOld.loc[c])
                classdist.append(dist)
            dataclast_rls['clust'][r] = int(np.array(classdist).argmin()+1)
        for r in range(1, clusnum +1):
            for f in features:
                centroidsnew[f][r] = dataclast_rls[dataclast_rls.clust == r][f].mean()
        centroidsOld = centroidsnew
        iteration += 1
```

K-means SkLearn

In [ ]:
```
# k = 3
from sklearn.cluster import KMeans
import numpy as np
kmeans3 = KMeans(n_clusters=3, init = 'random', n_init = 10,  random_state=0).fit(dataclast_rls[features])
kmeans3.labels_
kmeans3.cluster_centers_
```

In [ ]:
```
# k = 5
from sklearn.cluster import KMeans
import numpy as np
kmeans5 = KMeans(n_clusters=3, init = 'random', n_init = 10,  random_state=0).fit(dataclast_rls[features])
kmeans5.labels_
kmeans5.cluster_centers_
```

Appendix B. Code example for computing relative distances.

Cluster interpretation

Relative distances

k = 5

In [103]:
```
# Cluster № 1
reldis5_1 = pd.DataFrame(index = ['ckv', 'cv','difference', 'dkv'], columns = features)
for i in features:
    reldis5_1[i]['ckv'] = dataclast[dataclast.clust5 == 1][i].mean()
    reldis5_1[i]['cv'] = dataclast[i].mean()
    reldis5_1[i]['difference'] = reldis5_1[i]['ckv'] - reldis5_1[i]['cv']
    reldis5_1[i]['dkv'] = 100*((reldis5_1[i]['ckv']/reldis5_1[i]['cv'])-1)
reldis5_1
```

Out[103]:

	u	g	r	i	z	redshift
ckv	16.8654	15.4638	14.9278	14.7145	14.5574	0.0148811
cv	18.6194	17.3719	16.841	16.5836	16.4228	0.143726
difference	-1.75399	-1.90817	-1.91316	-1.86911	-1.86548	-0.128845
dkv	-9.42026	-10.9842	-11.3601	-11.2709	-11.3591	-89.6462

In [104]:
```
# Cluster № 2
reldis5_2 = pd.DataFrame(index = ['ckv', 'cv','difference', 'dkv'], columns = features)
for i in features:
    reldis5_2[i]['ckv'] = dataclast[dataclast.clust5 == 2][i].mean()
    reldis5_2[i]['cv'] = dataclast[i].mean()
    reldis5_2[i]['difference'] = reldis5_2[i]['ckv'] - reldis5_2[i]['cv']
    reldis5_2[i]['dkv'] = 100*((reldis5_2[i]['ckv']/reldis5_2[i]['cv'])-1)
reldis5_2
```

Out[104]:

	u	g	r	i	z	redshift
ckv	19.1574	18.1036	17.6774	17.457	17.3415	0.0618462
cv	18.6194	17.3719	16.841	16.5836	16.4228	0.143726
difference	0.538025	0.731651	0.8364	0.873437	0.918709	-0.0818795
dkv	2.8896	4.21168	4.96646	5.26688	5.5941	-56.9693

Appendix C. Code example for computing linear regression

Linear regression

In [381]:

```
datareg = data[['u','z']].copy()

datareg_train = datareg[:int(len(datareg)/2)]
datareg_test = datareg[-int(len(datareg)/2):]

datareg_z_train = datareg.z[:int(len(datareg)/2)]
datareg_z_test = datareg.z[-int(len(datareg)/2):]

regr = linear_model.LinearRegression()

regr.fit(datareg_train, datareg_z_train)

datareg_z_pred = regr.predict(datareg_test)
```