FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTION

OF HIGHER EDUCATION

NATIONAL RESEARCH UNIVERSITY

HIGHER SCHOOL OF ECONOMICS

Faculty of Computer Sciences

Department of Data Analysis and Artificial Intelligence

Educational program "Data Science"

MODERN DATA ANALYSIS
HOMEWORK 2020

Govorova Irina

Telnov Sergey

Shenker Anastasia

Yakovlev Daniil

Moscow, 2020

CONTENTS

DATASET DESCRIPTION **3**

K-MEANS CLUSTERING **6**

RELATIVE DIFFERENCE 7

QUETELET INDECES 8

MAKING SETS $\boldsymbol{V+}$ AND $\boldsymbol{V-}$. 10

SETS FOR 4-CLUSTER SPLIT. 10

CONCEPTUALIZATION 16

BOOTSTRAP **18**

CONTINGENCY TABLE **20**

PCA/SVD **26**

STANDARDIZING DATA 26

SVD 26

HIDDEN RANKING FACTOR 30

DATA VISUALIZATION 33

CORRELATION COEFFICIENT **36**

LINEAR REGRESSION 38

CORRELATION COEFFICIENTS 39

DETERMINACY COEFFICIENTS 40

PREDICTIONS 41

APPENDIX **42**

K-MEANS 42

BOOTSTRAP 47

CONTIGENCY TABLE 48

PCA SVD 52

CORRELATION COEFFICIENT 58

Dataset description

For the implementation of this project, a "Star dataset" was chosen, since it approaches well for the task of clustering, represents an interesting topic and gives a possibility to explore some enthralling issues.

The source of the "Star dataset" is a system for organizing competitions for data research and a social network for data processing and machine learning specialists Kaggle[1].

"Star dataset" consists of 241 observations and the following features:

- `Temperature (quantitative)`
- `Luminosity (quantitative)`
- `Radius (quantitative)`
- `Absolute Magnitude (quantitative)`
- `Star Type (categorical)`
- `Star Color (categorical)`
- `Spectral Class (categorical)`

Temperature

"Temperature" is an absolute temperature of the star measured in Kelvins.

Luminosity

"Luminosity" is a relative luminosity of a star, which is calculated as $\frac{L}{L_0}$, where L is a luminosity of a star and $L_0 = 3.828 \cdot 10^{26}$ Watts is an average luminosity of the Sun. Luminosity of the star is the total amount of electromagnetic energy emitted by a star per unit of time. The luminosity of the main sequence stars can be approximately calculated by the formula:

$$L = 4\pi R^2 \cdot \sigma T^4$$

R – star radius

T – star photosphere temperature

σ – Stefan - Boltzmann constant

[1] https://www.kaggle.com/deepu1109/star-dataset?select=6+class+csv.csv

Radius

"Radius" is a relative radius of a star, which is calculated as $\frac{R}{R_0}$, where R is a radius of a star and $R_0 = 6.9551 \cdot 10^8$ m is an average radius of the Sun.

Absolute Magnitude

"Absolute Magnitude" is a value of a star absolute magnitude measured in Mv. Absolute magnitude is a measure of the luminosity of a celestial object. For stars it is defined as the apparent magnitude of an object if it was located at a distance of 10 parsecs from the observer and would not experience any interstellar or atmospheric absorption.

Star Type

"Star Type" is for one of the following types: Red Dwarf, Brown Dwarf, White Dwarf, Main Sequence, Supergiants, Hypergiants.

A red dwarf is the smallest and coolest kind of star on the main sequence. Red dwarfs are by far the most common type of star in the Milky Way, at least in the neighborhood of the Sun, but because of their low luminosity, individual red dwarfs cannot be easily observed.

A brown dwarf is a type of substellar object that has a mass between the most massive gas giant planets and the least massive stars.

A white dwarf is a stellar core remnant composed mostly of electron-degenerate matter. A white dwarf is very dense: its mass is comparable to that of the Sun, while its volume is comparable to that of Earth. A white dwarf's faint luminosity comes from the emission of stored thermal energy.

Main-sequence stars or dwarf stars are the most numerous true stars in the universe, and include the Earth's Sun. The main sequence is a continuous and distinctive band of stars that appears on plots of stellar color versus brightness.

Supergiants are among the most massive and most luminous stars.

A hypergiant is a very rare type of star that has an extremely high luminosity, mass, size and mass loss because of their extreme stellar winds.

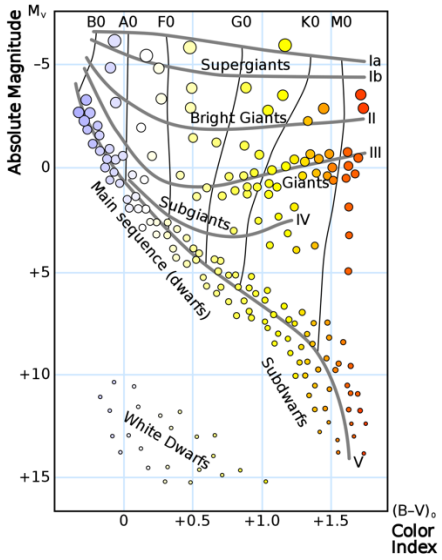

Fig. 1. The Hertzsprung–Russell diagram relates stellar classification with absolute magnitude, luminosity, and surface temperature.

Star Color

"Star Color" is a color of a star, which depends on its temperature and can be White, Red, Blue, Blue-White, Yellow, Yellow-Orange, Orange-Red, etc.

Spectral Class

"Spectral Class" is one of the following spectral type: O, B, A, F, G, K, M according to the Morgan–Keenan (MK) classification system from the hottest (O type) to the coolest (M type).

K-Means clustering

For K-Means clustering all 4 quantitative features were considered:

- «Temperature»
- «Luminosity»
- «Radius»
- «Absolute magnitude»

We chose all these features because they represent different characteristics of stars and they are not linearly dependent. All these four features are equally significant for category definition.

We standardized all columns and then applied K-Means algorithm to split the data into 4 clusters and into 7 clusters. K-Means was applied 10 times with following parameters: maximum number of iterations 500, random initial cluster centers, relative tolerance 0.0001, EM-style algorithm and different random state, which equals the number of iteration. From 10 results of clustering we chose the best according to the parameter «inertia»: the less inertia, the better clustering.

In the following table, you can see results of K-Means initializations, where «i» is a number of initialization and a value of random state.

Table 1. Results of K-Means initializations

	i = 0	**i = 1**	**i = 2**	**i = 3**	**i = 4**	**i = 5**	**i = 6**	**i = 7**	**i = 8**	**i = 9**
4 clusters	281.5	257.21	283.93	277.05	264.86	282.05	281.51	281.51	257.21	256.9
7 clusters	140.63	147.51	136.51	137.01	137.01	137.01	198.68	137.06	192.15	137.01

The best result for 4 clusters was received in the initialization with i = 9. The best result for 7 clusters was received in the initialization with i = 2.

There is a more thorough interpretation of all four clusters is given below.

Relative difference

In this section, relative difference for all clusters and all quantitative features are counted. Relative difference shows how much cluster mean of a feature differs from grand mean of the same feature.

In the following table, you can see relative differences for every feature and for every cluster of 4-cluster K-Means.

Table 2. Relative differences for every feature and for every cluster of 4-cluster K-Means

	Temperature	**Luminosity**	**Radius**	**Absolute magnitude**
Relative difference for cluster “0”	-53.53	134.18	476.36	-327.05
Relative difference for cluster “1”	171.34	423.36	337.89	-288.69
Relative difference for cluster “2”	-39.18	-99.99	-99.85	192.67
Relative difference for cluster “3”	71.66	59.37	-81.71	-201.15

In the following table, you can see relative differences for every feature and for every cluster of 7-cluster K-Means.

Table 3. Relative differences for every feature and for every cluster of 7-cluster K-Means

	Temperature	**Luminosity**	**Radius**	**Absolute magnitude**
Relative difference for cluster “0”	-64.35	-100	-99.91	241.32
Relative difference for cluster “1”	-9.93	-84.26	-96.54	-94.11
Relative difference for cluster “2”	-51.24	130.11	467.12	-325.33
Relative difference for cluster “3”	171	56.89	-88.37	-213.82
Relative difference for cluster “4”	2.32	216.21	-76.95	249.08
Relative difference for cluster “5”	196.75	390.51	449.73	-299.48
Relative difference for cluster “6”	53.71	-100	-100	172.46

The absolute value of relative difference in these tables shows the percent of deviation from the grand mean of the feature. The greater the value, the greater the deviation. The sign shows the direction of deviation. Minus means that the mean of

the cluster is less than the grand mean and plus means that the mean of the cluster is greater than the grand mean.

Quetelet indeces

Quetelet index shows what part of category belongs to a cluster. It doesn't show an exact part of category but show it comparably: the less absolute value the bigger part of cluster belongs to a category.

Quetelet index is calculated by the formula:

$$q_{kv} = 100\left[\frac{p_{kv}}{p_k p_v} - 1\right],$$

where p_{kv} – the proportion of objects belonging to a cluster and a category simultaneously, p_k – the proportion of objects belonging to a cluster, p_v – the proportion of objects belonging to a category.

In the following tables, you can see quetelet indeces for each category of features «Star type», «Star color» and «Star Class» for 4-cluster splitting.

Table 4. Quetelet indeces for star types for 4-cluster K-Means

	0 – Red Dwarf	**1 – Brown Dwarf**	**2 – White Dwarf**	**3 – Main Sequence**	**4 – SuperGiants**	**5 – HyperGiants**
Quetelet index for cluster "0"	-100	-100	-100	-100	-100	-100
Quetelet index for cluster "1"	-100	-100	-100	-100	71.43	328.57
Quetelet index for cluster "2"	77.78	77.78	64.44	-20	-100	-100
Quetelet index for cluster "3"	-100	-100	-70.97	112.9	248.39	-90.32

Table 5. Quetelet indeces for star colors for 4-cluster K-Means

	Red	**Blue White**	**White**	**Yellowish White**	**Blue white**	**Pale yellow orange**	**Blue**
Quetelet index for cluster "0"	69.955	-100	136.45	-100	-100	-100	-84.95
Quetelet index for cluster "1"	-100	-100	-100	-100	-100	-100	211.69
Quetelet index for cluster "2"	26.98	60	26.98	77.78	77.78	77.78	-67.68
Quetelet index for cluster "3"	-68.89	-61.29	-100	-100	-100	-100	139.3

	Blue-white	Whitish	yellow-white	Orange	White-Yellow	white	Blue
Quetelet index for cluster “0”	-68.17	-100	-100	727.59	-100	-100	-100
Quetelet index for cluster “1”	163.74	-100	-100	-100	-100	-100	-100
Quetelet index for cluster “2”	-79.49	77.78	77.78	-100	77.78	77.78	77.78
Quetelet index for cluster “3”	167.99	-100	-100	-100	-100	-100	-100

	yellowish	Yellowish	Orange-Red	Blue white	Blue-White
Quetelet index for cluster “0”	-100	-100	-100	-100	-100
Quetelet index for cluster “1”	-100	-100	-100	-100	-100
Quetelet index for cluster “2”	77.78	77.78	77.78	77.78	77.78
Quetelet index for cluster “3”	-100	-100	-100	-100	-100

Table 6. Quetelet indeces for spectral classes for 4-cluster K-Means

	M	B	A	F	O	K	G
Quetelet index for cluster “0”	64.03	-82.01	-12.89	-100	-79.31	175.86	727.59
Quetelet index for cluster “1”	-100	86.34	-100	-100	285.71	-100	-100
Quetelet index for cluster “2”	28.13	-14.98	12.28	77.78	-100	18.52	-100
Quetelet index for cluster “3”	-68.61	51.47	1.87	-100	190.32	-100	-100

The value -100 means that no object belongs to a cluster and a category simultaneously.

The greater the number (the smaller the absolute number), the more objects from a cluster belong to a category simultaneously and the more objects from a category belong to a cluster.

The value 0 is possible only when all objects belong to one category and to one cluster.

These tables show the differences in clusters.

Making sets V^+ and V^-.

V^+ is a set, which is formed by features and categories, values of which is greater than 35%.

V^- is a set, which is formed by features and categories, values of which is less than -35%.

Sets for 4-cluster split.

Cluster «0»

V^+ for cluster 0 consist of following numeric features and categories: «Luminosity», «Radius», «HyperGiants», «Red», «White», «Orange, «M», «K», «G».

V^- for cluster 0 consist of following numeric features and categories: «Temperature», «Absolute magnitude», *«Red Dwarf», «Brown Dwarf», «White Dwarf», «SuperGiants»*, *«Blue White», «Yellowish White», «Blue white», «Pale yellow orange»*, «Blue», «Blue-white», *«Whitish», «yellow-white»*, «Orange», *«White-Yellow», «white», «Blue», «yellowish», «Yellowish», «Orange-Red», «Blue white», «Blue-White»*, «B», «F», «O».

Cluster «0» is less than average in these features. But we want to highlight that most color categories are equal -100 (features in italics), which means that there is no objects of such cluster and this is true for every cluster.

In general cluster «0» is very much greater in sizes and has red, orange and blue colors.

Cluster «1»

V^+ for cluster 1 consist of following numeric features and categories: «Temperature», «Luminosity», «Radius», «SuperGiants», «HyperGiants», «Blue», «Blue-white», «B», «O».

V^- for cluster 1 consist of following numeric features and categories: «Absolute magnitude», *«Red Dwarf», «Brown Dwarf», «White Dwarf», «Main*

Sequence», «Red», «Blue White», «White», «Yellowish White», «Blue white», «Pale yellow orange», «Whitish», «yellow-white», «Orange», «White-Yellow», «white», «Blue», «yellowish», «Yellowish», «Orange-Red», «Blue white», «Blue-White», «M», «A», «F», «K», «G».

In general cluster «1» is greater in sizes and has more blue and blue-white colors.

Cluster «2»

V^+ for cluster 2 consist of following numeric features and categories: «Absolute magnitude», «Red Dwarf», «Brown Dwarf», «White Dwarf», «Blue White», «Yellowish White», «Blue white», «Pale yellow orange», «Whitish», «yellow-white», «White-Yellow», «white», «Blue», «yellowish», «Yellowish», «Orange-Red», «Blue white», «Blue-White», «F».

V^- for cluster 2 consist of following numeric features and categories: «Luminosity», «Radius», *«SuperGiants», «HyperGiants»,* «Blue», «Blue-white», «Orange», *«O», «G».*

Cluster «2» is very much less in sizes and has more white, yellow and blue colors.

Cluster «3»

V^+ for cluster 3 consist of following numeric features and categories: «Temperature», «Luminosity», «Main Sequence», «SuperGiants», «Blue», «Blue-white», «B», «O».

V^- for cluster 3 consist of following numeric features and categories: «Radius», «Absolute magnitude», *«Red Dwarf», «Brown Dwarf»,* «White Dwarf», «HyperGiants», «Red», «Blue White», *«White», «Yellowish White», «Blue white», «Pale yellow orange», «Whitish», «yellow-white», «Orange», «White-Yellow», «white», «Blue», «yellowish», «Yellowish», «Orange-Red», «Blue white», «Blue-White»,* «M», *«F», «K», «G».*

Cluster «3» is greater in temperature and luminosity, has more blue colors, less in size and magnitude.

Sets for 7-cluster split.

In the following tables, you can see quetelet indeces for each category of features «Star type», «Star color» and «Star Class» for 7-cluster splitting.

Table 7. Quetelet indeces for star types for 7-cluster K-Means

	0 – Red Dwarf	1 – Brown Dwarf	2 – White Dwarf	3 – Main Sequence	4 – SuperGiants	5 – HyperGiants
Quetelet index for cluster “0”	-100	-100	-100	-100	-100	-100
Quetelet index for cluster “1”	-100	-100	-100	-100	71.43	328.57
Quetelet index for cluster “2”	77.78	77.78	64.44	-20	-100	-100
Quetelet index for cluster “3”	-100	-100	-70.97	112.9	248.39	-90.32
Quetelet index for cluster “4”	-100	-100	-100	-100	500	-100
Quetelet index for cluster “5”	-100	-100	-100	-100	-45.45	445.45
Quetelet index for cluster “6”	-100	-100	500	-100	-100	-100

Table 8. Quetelet indeces for star colors for 7-cluster K-Means

	Red	Blue White	White	Yellowish White	Blue white	Pale yellow orange	Blue
Quetelet index for cluster “0”	88.38	-73.63	50.71	-12.09	75.82	163.74	-100
Quetelet index for cluster “1”	-86.61	-100	-100	-100	-100	-100	-72.73
Quetelet index for cluster “2”	64.29	-100	128.57	-100	-100	-100	-85.45
Quetelet index for cluster “3”	-68.89	-61.29	-100	-100	-100	-100	139.3
Quetelet index for cluster “4”	-37.5	-100	-100	-100	-100	-100	209.9
Quetelet index for cluster “5”	-100	-100	-100	-100	-100	-100	177.69
Quetelet index for cluster “6”	-100	644.83	18.23	451.72	175.86	-100	80.56

	Blue-white	Whitish	yellow-white	Orange	White-Yellow	white	Blue
Quetelet index for cluster “0”	-100	-100	-100	-100	163.74	-12.09	-100
Quetelet index for cluster “1”	275	650	650	-100	-100	-100	-100
Quetelet index for cluster “2”	-38.46	-100	-100	700	-100	-100	-100
Quetelet index for cluster “3”	167.99	-100	-100	-100	-100	-100	-100
Quetelet index for cluster “4”	-100	-100	-100	-100	-100	-100	-100
Quetelet index for cluster “5”	235.66	-100	-100	-100	-100	-100	-100
Quetelet index for cluster “6”	-100	-100	-100	-100	-100	451.72	727.59

	yellowish	Yellowish	Orange-Red	Blue white	Blue-White
Quetelet index for cluster “0”	-100	-100	-100	-100	-100
Quetelet index for cluster “1”	650	650	650	-100	650
Quetelet index for cluster “2”	-100	-100	-100	-100	-100
Quetelet index for cluster “3”	-100	-100	-100	-100	-100
Quetelet index for cluster “4”	-100	-100	-100	-100	-100
Quetelet index for cluster “5”	-100	-100	-100	-100	-100
Quetelet index for cluster “6”	727.59	-100	-100	727.59	-100

Table 9. Quetelet indeces for spectral classes for 7-cluster K-Means

	M	B	A	F	O	K	G
Quetelet index for cluster “0”	90.08	-100	-2.83	-37.94	-100	-100	-100
Quetelet index for cluster “1”	-86.49	-2.17	294.74	252.94	-62.5	400	-100
Quetelet index for cluster “2”	58.56	-65.22	-15.79	-100	-80	166.67	700
Quetelet index for cluster “3”	-100	104.16	-100	-100	265.22	-100	-100
Quetelet index for cluster “3”	-36.94	-100	-100	-100	325	-100	-100
Quetelet index for cluster “3”	-100	137.15	-100	-100	227.27	-100	-100
Quetelet index for cluster “3”	-100	331.78	-100	143.41	-100	-100	-100

Cluster «0»

V^+ for cluster 0 consist of following numeric features and categories: «Absolute magnitude», «Red Dwarf», «Brown Dwarf», «Red», «White», «Blue white», «Pale yellow orange», «White-Yellow», «M»,

V^- for cluster 0 consist of following numeric features and categories: «Temperature», «Luminosity», «Radius», *«Main Sequence», «SuperGiants», «HyperGiants»*, «Blue White», *«Blue», «Blue-white», «Whitish», «yellow-white», «Orange», «Blue», «yellowish», «Yellowish», «Orange-Red», «Blue white», «Blue-White», «B», «O», «K», «G».*

Cluster «0» is less in sizes, temperature and luminosity, has more red, orange and white colors.

Cluster «1»

V^+ for cluster 1 consist of following numeric features and categories: «Main Sequence», «Blue-white», «Whitish», «yellow-white», «yellowish», «Yellowish», «Orange-Red», «Blue-White», «A», «F», «K».

V^- for cluster 1 consist of following numeric features and categories: «Luminosity», «Radius», «Absolute magnitude», *«Red Dwarf», «Brown Dwarf», «White Dwarf», «HyperGiants»,* «Red», *«Blue White», «White», «Yellowish White», «Blue white», «Pale yellow orange»*, «Blue», *«Orange», «White-Yellow», «white», «Blue», «Blue white»,* «M», «O», *«G».*

Cluster 1 has more size and more yellow colors.

Cluster «2»

V^+ for cluster 2 consist of following numeric features and categories: «Luminosity», «Radius», «HyperGiants», «Red», «White», «Orange», «M», «K», «G».

V^- for cluster 2 consist of following numeric features and categories: «Temperature», «Absolute magnitude», *«Red Dwarf», «Brown Dwarf», «White*

Dwarf», «Main Sequence», «SuperGiants», «Blue White», «Yellowish White», «Blue white», «Pale yellow orange», «Whitish», «yellow-white», «White-Yellow», «white», «Blue», «yellowish», «Yellowish», «Orange-Red», «Blue white», «Blue-White», «B», *«F»*.

Cluster 2 has much greater luminosity and size and more red and orange.

Cluster «3»

V^+ for cluster 3 consist of following numeric features and categories: «Temperature», «Luminosity», «Main Sequence», «SuperGiants», «Blue», «Blue-white», «B», «F».

V^- for cluster 3 consist of following numeric features and categories: «Radius», «Absolute magnitude», *«Red Dwarf», «Brown Dwarf», «White Dwarf», «HyperGiants»,* «Red», «Blue White», *«White», «Yellowish White», «Blue white», «Pale yellow orange», «Whitish», «yellow-white», «Orange», «White-Yellow», «white», «Blue», «yellowish», «Yellowish», «Orange-Red», «Blue white», «Blue-White», «M», «A», «O», «K», «G».*

Cluster 3 has much higher temperature and luminosity and more blue colors.

Cluster «4»

V^+ for cluster 4 consist of following numeric features and categories: «Luminosity», «SuperGiants», «Blue», «O».

V^- for cluster 4 consist of following numeric features and categories: «Radius», «Absolute magnitude», *«Red Dwarf», «Brown Dwarf», «White Dwarf», «Main Sequence», «HyperGiants»,* «Red», *«Blue White», «White», «Yellowish White», «Blue white», «Pale yellow orange», «Blue-white», «Whitish», «yellow-white», «Orange», «White-Yellow», «white», «Blue», «yellowish», «Yellowish», «Orange-Red», «Blue white», «Blue-White»,* «M», *«B», «A», «F», «K», «G».*

Cluster 4 has a very much greater size and luminosity and more blue colors.

Cluster «5»

V^+ for cluster 5 consist of following numeric features and categories: «Temperature», «Luminosity», «Radius», «HyperGiants», «Blue», «Blue-white», «B», «O».

V^- for cluster 5 consist of following numeric features and categories: «Absolute magnitude», *«Red Dwarf», «Brown Dwarf», «White Dwarf», «Main Sequence», «Red», «Blue White», «White», «Yellowish White», «Blue white», «Pale yellow orange», «Whitish», «yellow-white», «Orange», «White-Yellow», «white», «Blue», «yellowish», «Yellowish», «Orange-Red», «Blue white», «Blue-White», «M», «A», «F», «K», «G».*

Cluster 4 has a very much greater size, luminosity and temperature and more blue colors.

Cluster «6»

V^+ for cluster 6 consist of following numeric features and categories: «Temperature», «Absolute magnitude», «White Dwarf», «Blue White», «Yellowish White», «Blue white», «Blue», «white», «Blue», «Blue white»,

V^- for cluster 6 consist of following numeric features and categories: «Luminosity», «Radius», *«Red Dwarf», «Brown Dwarf», «Main Sequence», «SuperGiants», «HyperGiants», «Red», «Pale yellow orange», «Blue-white», «Whitish», «yellow-white», «Orange», «White-Yellow», «yellowish», «Yellowish», «Orange-Red», «Blue-White».*

Cluster 6 has less size, higher temperature and more blue and white colors.

Conceptualization

Now let us have a look on the results of clustering according to the types of stars. We will assume that a type of star belongs to a cluster, which contains most of the objects of this type.

In the following graph, we can see distribution of different types of stars for 4 clusters.

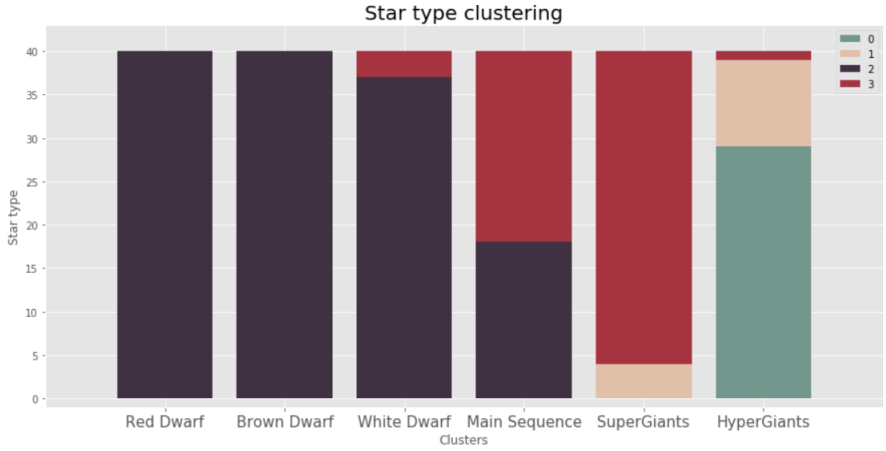

Fig. 2. Distribution of different types of stars for 4 clusters

According to our assumption, we can say that «Red Dwarf», «Brown Dwarf» and «White Dwarf» are in the cluster «2», «SuperGiants» and «HyperGiants» are in the cluster «3». «HyperGiants» are in the cluster «0». The cluster «1» contains some stars of «SuperGiants» type and some stars of «HyperGiants».

Certainly, objects are not splitted perfect. There are few objects of «WhiteDwarf» and «HyperGiants» belong to the cluster «3» and almost a half of objects of «Main Sequence» belong to the cluster «2».

In the following graph, we can see distribution of different types of stars for 4 clusters.

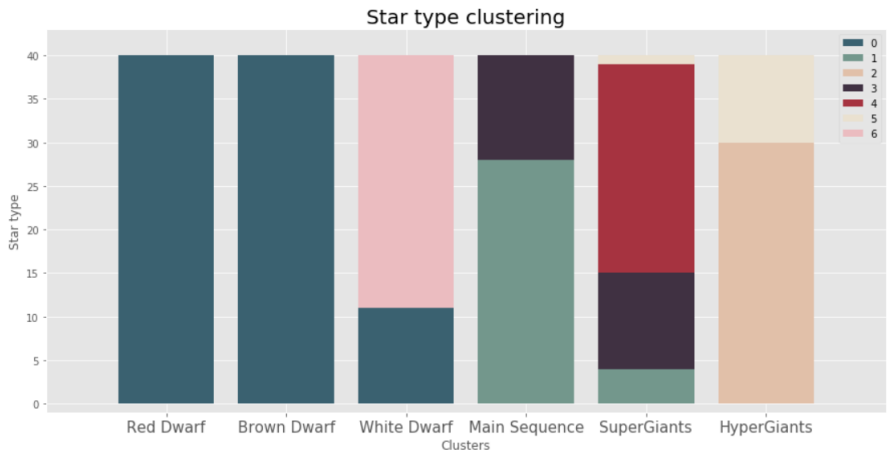

Fig. 3. Distribution of different types of stars for 4 clusters

According to our assumption, we can say that «Red Dwarf», «Brown Dwarf» are in the cluster «0», «White Dwarf» is in cluster «6», «Main Sequence» is in cluster «1» «SuperGiants» are in the cluster «4». «HyperGiants» are in the cluster «2». The cluster «5» contains some stars of «HyperGiants» and «SuperGiants» and cluster «3» contains some stars of «Main Sequence» and «SuperGiants».

K-Means in both cases (4 clusters and 7 clusters) split stars according to their type. We think that 4-cluster splitting is better. Because it is more interpretable with given features in dataset for people without astronomic education. It splits dwarf stars, stars with average / a little bit bigger size and giant stars, while 7-cluster splitting select more subtypes of stars which are not selected in usual classifications.

Also 4-cluster splitting cope with big number of color categories better. There are a lot of the same colors which are written differently (for example: «Blue White», «Blue-white», «Blue white», «Blue-White») but in fact they are the same, little number of clusters let to ignore this thing.

Bootstrap

For bootstrap the feature “Temperature” and 4-cluster partition with a number of initialization and a value of random state equal to 9 from previous part are used. For each application of bootstrap non-pivotal and pivotal versions are considered.

The first step is to find the 95% confidence interval for feature’s grand mean by using bootstrap.

As can be seen from the following table, the average temperature of the considered stars is between 9 206.58 K and 11 714.13 K for non-pivotal version and between 9 308.09 and 11 681.74 for pivotal version according to 95% confidence interval.

Table 10. Grand mean and 95% confidence intervals for "Temperature"

	Value
Grand mean	10 497.46
Non-pivotal 95% confidence interval	9 206.58 – 11 714.13
Pivotal 95% confidence interval	9 308.09 – 11 681.74

The second step is to take 2 clusters of the feature and for them to compare the within-cluster means using bootstrap. Cluster 0 and cluster 1 are considered.

As can be seen from the following table, the 95% confidence interval for the difference of within-cluster means is from 19 084.64 K to 27 937.23 K for non-pivotal version and from 19 186.28 K to 27 864.09 K for pivotal version.

Table 11. 95% confidence intervals for within-cluster means difference

	Value
Non-pivotal 95% confidence interval	19 084.64 – 27 937.23
Pivotal 95% confidence interval	19 186.28 – 27 864.09

So, the temperature in cluster 1 is in average at 23 510.935 K (23 525.23 K for pivotal version) higher than in cluster 0.

The next step is to compare the grand mean of the cluster with the within-cluster mean for the feature by using bootstrap. The grand mean is considered for the cluster 1.

As can be seen from the following table, the difference of the cluster 1 grand mean and within-cluster mean for the feature "Tenperature" is between 13 646.25 K and 22 495.87 K for non-pivotal version and between 13 577.53 K and 22 457.15 K for pivotal version.

Table 12. 95% confidence intervals for the difference between cluster 1 grand mean and within-cluster mean for the feature

	Value
Non-pivotal 95% confidence interval	13 646.25 – 22 495.87
Pivotal 95% confidence interval	13 577.53 – 22 457.15

So, the average temperature in cluster 1 is in average at K 18 071.04 (18 017.34 K for pivotal version) higher than the average temperature in other clusters.

Contingency Table

For contingency table 3 nominal features were considered. One of them, “Star Type”, was taken from categorical features of the data. Two others were made from quantitative features of the data.

To choose, which quantitative features to take, the histograms were builded for all four quantitative features: “Temperature”, “Luminosity”, “Radius”, “Absolute Magnitude”.

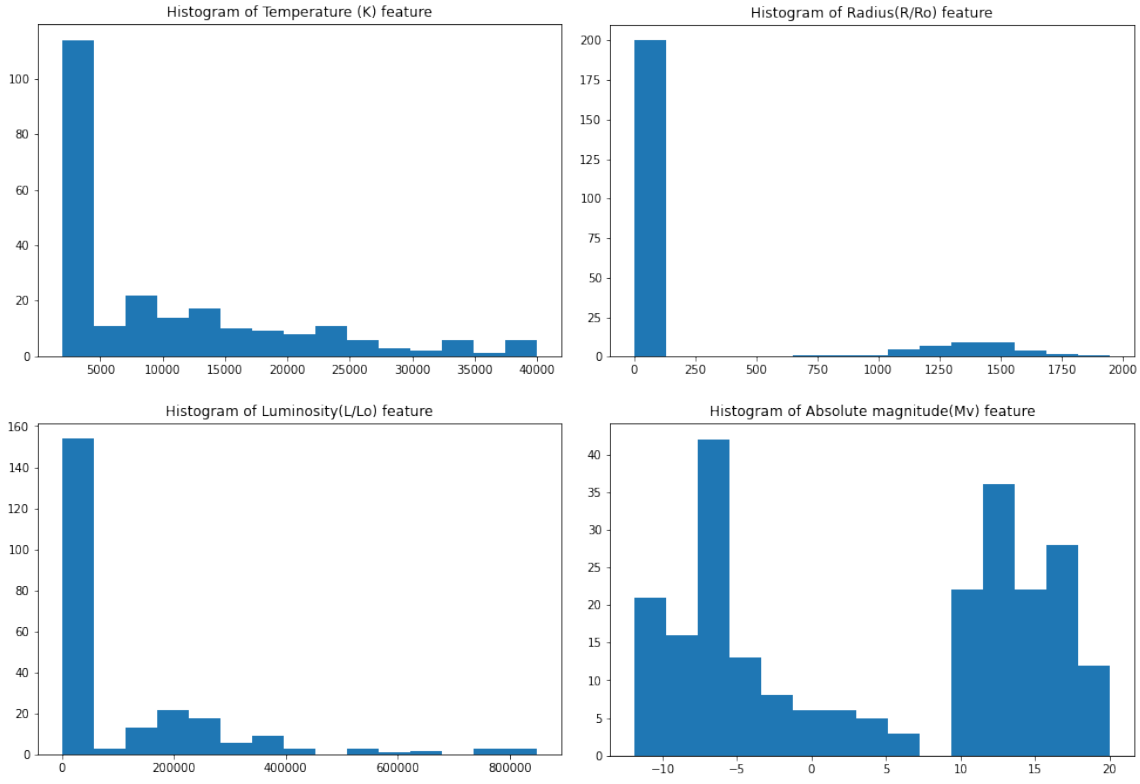

Fig. 4. Histograms of features

Two features, “Temperature” and “Absolute Magnitude” were taken according to their splitting ability.

To be used as nominal features, “Temperature” and “Absolute Magnitude” were splited into 6 blocks with the limits as in table below.

Table 13. Upper bounds for block splitting

Block number	Temperature	Absolute magnitude
0	3 000	-10
1	3 500	-5
2	4 500	5
3	10 000	12
4	15 000	15
5	40 000	20

So, there are three nominal features with five possible values in each. In the following table distribution of values across blocks in these variables is presented.

Table 14. Upper bounds for block splitting

Block number	Star type	Temperature	Absolute magnitude
0	40	31	21
1	40	45	61
2	40	39	35
3	40	36	37
4	40	30	48
5	40	59	38

Now let’s build contingency tables over these features. The first table is the conditional frequency table and the second one is Quetelet relative index table.

In the following tables, cross-classification for features “Star Type” and “Temperature” is presented.

Table 15. Conditional frequency table for "Star Type" and "Temperature"

Star type / Temperature	0	1	2	3	4	5	Total
0	24	6	0	0	1	0	31
1	13	23	0	0	4	5	45
2	3	11	0	1	5	19	39
3	0	0	11	16	6	3	36
4	0	0	15	6	7	2	30
5	0	0	14	17	17	11	59
Total	40	40	40	40	40	40	240

Table 16. Quetelet relative index table for "Star Type" and "Temperature"

Star type / Temperature	0	1	2	3	4	5	Total
0	0.1	0.025	0	0	0.004167	0	0.129167
1	0.054167	0.095833	0	0	0.016667	0.020833	0.1875
2	0.0125	0.045833	0	0.004167	0.020833	0.079167	0.1625
3	0	0	0.045833	0.066667	0.025	0.0125	0.15
4	0	0	0.0625	0.025	0.029167	0.008333	0.125
5	0	0	0.058333	0.070833	0.070833	0.045833	0.245833
Total	0.166667	0.166667	0.166667	0.166667	0.166667	0.166667	1

According to these tables we can see, that the largest number of objects is at the intersection of zero blocks of two features, which correspond to the coolest Red Dwarf star type and the temperature below 3 000 K. It means, that the 60% of Red Dwarfs are below 3 000 K, and 77% of stars with a temperature below 3 000 K are Red Dwarfs.

It can be assumed that each next star type will correspond to a higher temperature level. And from the table we can see, that such a relation actually exists, although it is not strict enough. Near-diagonal elements have higher probability, than those that are far from the diagonal and zero blocks can be seen at the edges. The exception is for 5-th blocks, which means that there are Hypergiants with almost

equal probability can be from 4 500 K to 40 000 K or that the hottest stars can be predominantly Brown Dwarfs and Hypergiants.

These conclusions can be graphically confirmed by a following heatmap.

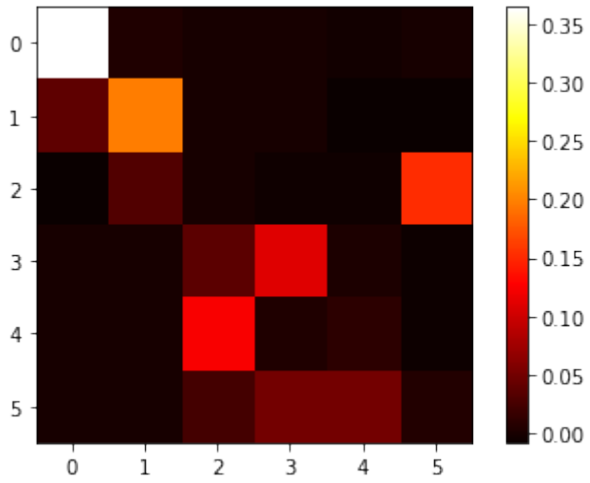

Fig. 5. Heatmap for "Star Type" and "Temperature" contingency table

In the following tables, cross-classification for features "Star Type" and "Absolute Magnitude" is presented.

Table 17. Conditional frequency table for "Star Type" and "Absolute Magnitude"

Star type **Absolute Magnitude**	**0**	**1**	**2**	**3**	**4**	**5**	**Total**
0	2	0	0	0	0	19	21
1	0	0	0	0	40	21	61
2	0	0	0	35	0	0	35
3	0	17	15	5	0	0	37
4	0	23	25	0	0	0	48
5	38	0	0	0	0	0	38
Total	40	40	40	40	40	40	240

Table 18. Quetelet relative index table for "Star Type" and "Absolute Magnitude"

Star type Absolute Magnitude	0	1	2	3	4	5	Total
0	0.008333	0	0	0	0	0.079167	0.0875
1	0	0	0	0	0.166667	0.0875	0.254167
2	0	0	0	0.145833	0	0	0.145833
3	0	0.070833	0.0625	0.020833	0	0	0.154167
4	0	0.095833	0.104167	0	0	0	0.2
5	0.158333	0	0	0	0	0	0.158333
Total	0.166667	0.166667	0.166667	0.166667	0.166667	0.166667	1

According to the table, there is inverse relationship between star type and absolute magnitude. This relation is stricter than for a previous pair, as we can clearly see the diagonal bordered by zero blocks. That means that the smaller and the colder the star, the brighter it is, as a lower value of absolute magnitude corresponds to a higher luminosity.

There is a little exception for zero blocks, which indicates that Red Dwarfs can rarely be as bright as Hypergiants despite its smallness and coldness.

These conclusions can be graphically confirmed by a following heatmap.

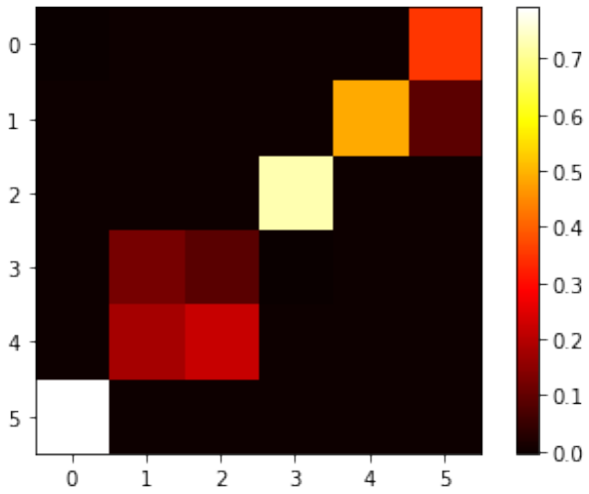

Fig. 6. Heatmap for "Star Type" and "Absolute Magnitude" contingency table

Observed values of contingency table may deviate from the hypothesis of statistical independence because of sampling bias. To evaluate how great the bias is the chi-squared as the average Quetelet index is used. So, it was calculated for both tables.

For the contingency table for “Star Type” and “Temperature” the following values were gained:

$$\text{Average Quetelet index} = 1.1740301610333046,$$

$$\chi^2 = 1.1702801610333051.$$

For the contingency table for “Star Type” and “Absolute Magnitude” the following values were gained:

$$\text{Average Quetelet index} = 3.0612300752157307,$$

$$\chi^2 = 3.16133676685686.$$

[Interpretation]

Under the hypothesis of independence, let’s find the 95% confidence interval that $N \cdot \chi^2$ is less than t = 23.685 (tabular value).

For “Star Type” and “Temperature”, $\chi^2 = 1.1703$. We have N = 241, which is greater than $\frac{23.685}{1.1703} = 20.2$. That is, at any N > 21 the hypothesis of statistical independence should be rejected at 95% confidence level.

For “Star Type” and “Absolute Magnitude”, $\chi^2 = 3.1613$. We have N = 241, which is greater than $\frac{23.685}{3.1613} = 7.4921$. That is, at any N > 8 the hypothesis of statistical independence should be rejected at 95% confidence level.

Similarly, for the 99% confidence that χ^2 is less than t = 32 (tabular value).

For “Star Type” and “Temperature”, $\chi^2 = 1.1703$. We have N = 241, which is greater than $\frac{32}{1.1703} = 27.3$. That is, at any N > 28 the hypothesis of statistical independence should be rejected at 99% confidence level.

For “Star Type” and “Absolute Magnitude”, $\chi^2 = 3.1613$. We have N = 241, which is greater than $\frac{32}{3.1613} = 10.122$. That is, at any N > 11 the hypothesis of statistical independence should be rejected at 95% confidence level.

PCA/SVD

For PCA and SVD the same features as for K-Means were chosen: “Temperature”, “Luminosity”, “Radius”, “Absolute Magnitude”, as they represent different linearly independent characteristics, all of which are important.

Standardizing data

We made three standardizations of the data: z-scoring, range and rank. Z-scoring is the same as StandaradScaler from sklearn.preprocessing module. It transforms feature into equivalent with 0 as the mean and 1 as standard deviation. Range standardization is done by substracting the mean (as in z-scoring standardization) and dividing by range (difference between maximum and minimum of the feature). Rank standardization is done by substracting mean and dividing by range. Rank standardization image a feature to a segment [0,1] and it is an equivalent of MinMaxScaler from sklearn.preprocessing module.

SVD

We made SVD for original data and for standardized datasets. We cannot show whole matrices of SVD because their shape is not appropriate for this. But in the following screenshots you can see these matrices partially.

SVD on original data:

```
SVD
U: [[ 3.91972718e-05 -1.75769413e-02 -8.99915824e-04 ...  4.32169233e-02
   4.21049371e-02  1.64754571e-01]
 [ 3.88645000e-05 -1.74279859e-02 -8.89844263e-04 ... -1.75505225e-01
  -1.25519068e-01  1.00027540e-02]
 [ 3.32174398e-05 -1.48957226e-02 -7.53627895e-04 ...  7.36611090e-02
   5.94190503e-02  1.88707634e-01]
 ...
 [ 1.66081594e-01  7.68328498e-02 -7.03178257e-02 ... -9.60287711e-01
   2.83796643e-02  1.56399942e-02]
 [ 1.25156666e-01  4.30850083e-02 -5.95435180e-02 ...  2.83465548e-02
  -9.78944404e-01  1.67258531e-02]
 [ 9.15455888e-02 -1.47110341e-01 -1.94199041e-01 ...  1.65308022e-02
   1.66469061e-02 -9.31105108e-01]]
Sigma: [3.23575983e+06 1.74397555e+05 6.86024479e+03 1.54256782e+02]
V*: [[ 4.13398236e-02  9.99143699e-01  1.69918151e-03 -1.90918876e-05]
 [-9.99143144e-01  4.13363114e-02  2.04829346e-03 -3.02725394e-04]
 [-1.97746978e-03  1.78251192e-03 -9.99988764e-01  3.92234827e-03]
 [-2.93922696e-04  2.45976481e-05  3.92298707e-03  9.99992262e-01]]
```

Fig. 7. SVD on original data

SVD on Z-scored standardized data:

```
SVD
U: [[ 0.06132141  0.02284989 -0.01668461 ...  0.01864612  0.0138139
  -0.17797746]
 [ 0.06246649  0.02289292 -0.01710579 ... -0.17109863 -0.12027931
   0.00577292]
 [ 0.06797651  0.02496603 -0.02031495 ... -0.06176361 -0.05318097
  -0.21918571]
 ...
 [-0.13339922  0.09919886 -0.05945816 ... -0.96444296  0.02338664
   0.01509854]
 [-0.10608223  0.07437359 -0.03232511 ...  0.02294347 -0.98285746
   0.00591266]
 [-0.15327085 -0.04132412  0.20715114 ...  0.01446281  0.00600947
  -0.90531603]]
Sigma: [24.07137206 15.02023728  9.46385723  8.08683657]
V*: [[-0.35018343 -0.55933789 -0.47477107  0.58232734]
 [-0.82161262 -0.00452564  0.56898115 -0.03453503]
 [ 0.37607837 -0.75509623  0.53314117 -0.06446119]
 [ 0.24675763  0.34198002  0.40818302  0.80965855]]
```

Fig. 8. SVD on Z-scored standardized data

SVD on range standardized data:

```
SVD
U: [[ 0.06566131  0.02565575  0.02068869 ...  0.01864612  0.0138139
  -0.17797746]
 [ 0.06735571  0.02584818  0.02427366 ... -0.17109863 -0.12027931
   0.00577292]
 [ 0.07515804  0.02852901  0.03827494 ... -0.06176361 -0.05318097
  -0.21918571]
 ...
 [-0.12373692  0.09462852  0.03597184 ... -0.96444296  0.02338664
   0.01509854]
 [-0.10530785  0.06975847 -0.0080703  ...  0.02294347 -0.98285746
   0.00591266]
 [-0.14378221 -0.03725076  0.25980513 ...  0.01446281  0.00600947
  -0.90531603]]
Sigma: [6.60736293 3.83106847 2.4030212  2.09064323]
V*: [[-0.29874304 -0.39982598 -0.46366863  0.73205408]
 [-0.81632001 -0.03596368  0.5763485   0.01227448]
 [ 0.46959659 -0.01569408  0.65145473  0.59568404]
 [-0.1544451   0.91575078 -0.16864323  0.330313  ]]
```

Fig. 9. SVD on range standardized data

SVD on rank standardized data:

```
SVD
U: [[ 0.08623102  0.03685018 -0.03085358 ...  0.0370722   0.03770005
   0.1709604 ]
 [ 0.08767107  0.03759248 -0.03159861 ... -0.17877728 -0.12768141
   0.01040809]
 [ 0.09371058  0.04164006 -0.03644315 ...  0.07046201  0.05906135
   0.18897713]
 ...
 [ 0.02093518 -0.1395772  -0.11346408 ... -0.96095803  0.02867502
   0.01436116]
 [ 0.01696782 -0.11252599 -0.07887478 ...  0.02867547 -0.97867616
   0.01492314]
 [ 0.05139389 -0.19891292 -0.00734302 ...  0.01461407  0.01502313
  -0.92038504]]
Sigma: [9.73832479 6.05617744 3.77203748 2.14904906]
V*: [[ 0.29448451  0.09911757  0.07199447  0.9477718 ]
 [-0.56235793 -0.52845282 -0.57413342  0.27360921]
 [ 0.69253414 -0.05494345 -0.7021478  -0.1560966 ]
 [ 0.342679   -0.84136469  0.41494106 -0.05000475]]
```

Fig. 10. SVD on rank standardized data

The goal of SVD is to find contributions of each component. In the following table you can see natural contribution of each feature in original dataset and in standardized datasets:

Table 19. Natural contribution of each feature

	Temperature	**Luminosity**	**Radius**	**Absolute magnitude**
Original data	1.0407e+13	3.0415e+10	4.7063e+07	2.3752e+04
Ranked std data	94.835	36.6773	14.2283	4.6184

Ranged std data	43.6573	14.677	5.7745	4.3708
Z-Scored std data	579.431	225.6075	89.5646	65.3969

It shows absolute value of contribution.

In the following table you can see percent contribution of each feature in datasets:

Table 20. Percent contribution of each feature

	Temperature	**Luminosity**	**Radius**	**Absolute magnitude**
Original data	99.71	0.2897	0.000448	0.000000227
Ranked std data	63.07	24.39	9.46	3.072
Ranged std data	63.75	21.43	8.43	6.38
Z-Scored std data	60.36	23.5	9.33	6.81

The first component («Temperature») contributes the most in each dataset.

In the following table you can see data scatter of each dataset:

Table 21. Data scatter

	Data Scatter
Original data	10 500 603 295 793.56
Ranked std data	150.359
Ranged std data	68.48
Z-Scored std data	960

Range standardized data has the least data scatter. Original data has the biggest data scatter because it is not standardized at all, each feature has values of different orders of magnitude.

Hidden ranking factor

Hidden factor is a value which characterize an object according to original features. Hidden factor is a value, which is highly correlated with original features.

Hidden ranking factor is rank standardized hidden factor. It has value from 0 to 100.

In the following tables you can see stars with top 5 hidden ranking factors for original data.

Table 22. Stars with top 5 hidden ranking factors for original data

	Temperature	Luminosity	Radius	Absolute magnitude	Star type	Star color	Spectral Class
233	27 739	849 420.0	1 252.0	-7.590	5	Blue-white	B
236	30 839	834 042.0	1 194.0	-10.630	5	Blue	O
101	40 000	813 000.0	14.0	-6.230	4	Blue	O
227	10 930	783 930.0	25.0	-6.224	4	Blue	O
229	21 738	748 890.0	92.0	-7.346	4	Blue	O

Their hidden ranking factors are in the following table.

Table 23. Top 5 hidden ranking factors for original data

	Hidden ranking factor
233	1
236	0.982
101	0.958
227	0.922
229	0.882

Contribution of this hidden factor is 99.7%

In the following tables you can see stars with top 5 hidden ranking factors for Z-score standardized data.

Table 24. Stars with top 5 hidden ranking factors for Z-score standardized data

	Temperature	Luminosity	Radius	Absolute magnitude	Star type	Star color	Spectral Class
4	1939	0.000138	0.1030	20.06	0	Red	M
122	3218	0.000540	0.1100	20.02	0	Red	M
128	2856	0.000896	0.0782	19.56	0	Red	M
188	2778	0.000849	0.1120	19.45	0	Red	M
121	3531	0.000930	0.0976	19.94	0	Red	M

Their hidden ranking factors are in the following table.

Table 25. Top 5 hidden ranking factors for Z-score standardized data

	Hidden ranking factor
4	1
122	0.992
128	0.9908
188	0.9903
121	0.9902

Contribution of this hidden factor is 60.36%

In the following tables you can see stars with top 5 hidden ranking factors for range standardized data.

Table 26. Stars with top 5 hidden ranking factors for range standardized data

	Temperature	Luminosity	Radius	Absolute magnitude	Star type	Star color	Spectral Class
4	1939	0.000138	0.1030	20.06	0	Red	M
122	3218	0.000540	0.1100	20.02	0	Red	M
121	3531	0.000930	0.0976	19.94	0	Red	M
128	2856	0.000896	0.0782	19.56	0	Red	M

125	3225	0.000760	0.1210	19.63	0	Red	M

Their hidden ranking factors are in the following table.

Table 27. Top 5 hidden ranking factors for range standardized data

	Hidden ranking factor
4	1
122	0.994
121	0.991
128	0.989
4	1

Contribution of this hidden factor is 63.75%

In the following tables you can see stars with top 5 hidden ranking factors for rank standardized data.

Table 28. Stars with top 5 hidden ranking factors for rank standardized data

	Temperature	Luminosity	Radius	Absolute magnitude	Star type	Star color	Spectral Class
121	3531	0.000930	0.0976	19.94	0	Red	M
122	3218	0.000540	0.1100	20.02	0	Red	M
4	1939	0.000138	0.1030	20.06	0	Red	M
125	3225	0.000760	0.1210	19.63	0	Red	M
128	2856	0.000896	0.0782	19.56	0	Red	M

Their hidden ranking factors are in the following table.

Table 29. Top 5 hidden ranking factors for rank standardized data

	Hidden ranking factor
121	1
122	0.9999

4	0.9898
125	0.9866
128	0.9809

Contribution of this hidden factor is 63.07%

Ranks of the same object from different dataset are not equal. Hidden factors of standardized datasets are quite similar. Objects of M class, red color, and 0 star type has the highest hidden factors. In the original dataset stars of other types have the highest hidden factors.

Data visualization

Using two first components (principal components) we made data visualization in coordinates of these two components.

Objects of different colors belong to different clusters, labels of clusters are mentioned in graph legend.

In the following graphs you can see visualizations of PCAs of Z-scored standardized data and range standardized data:

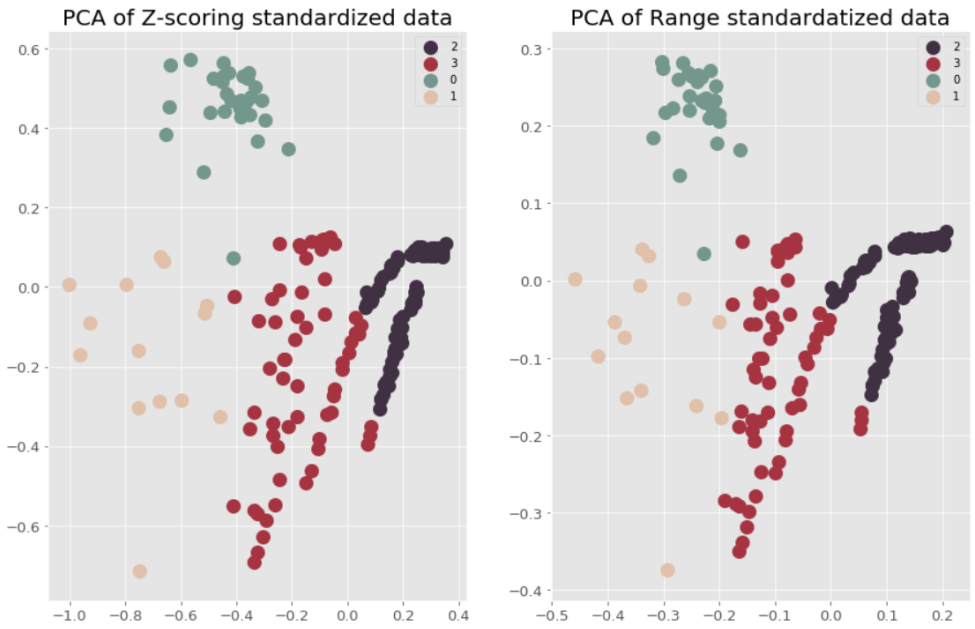

Fig. 11. PCAs of Z-scored standardized (left) and range standardized (right) data

Visualizations are quite similar, they just differ a bit in scale. Clusters, which were obtained by K-Means are well separated in both graphs. Also, PCAs have different axes scales.

After that we applied conventional PCA for original data and for Z-score standardized data. Also we applied PCA from sklearn.decomposition module.

Now let's compare the results.

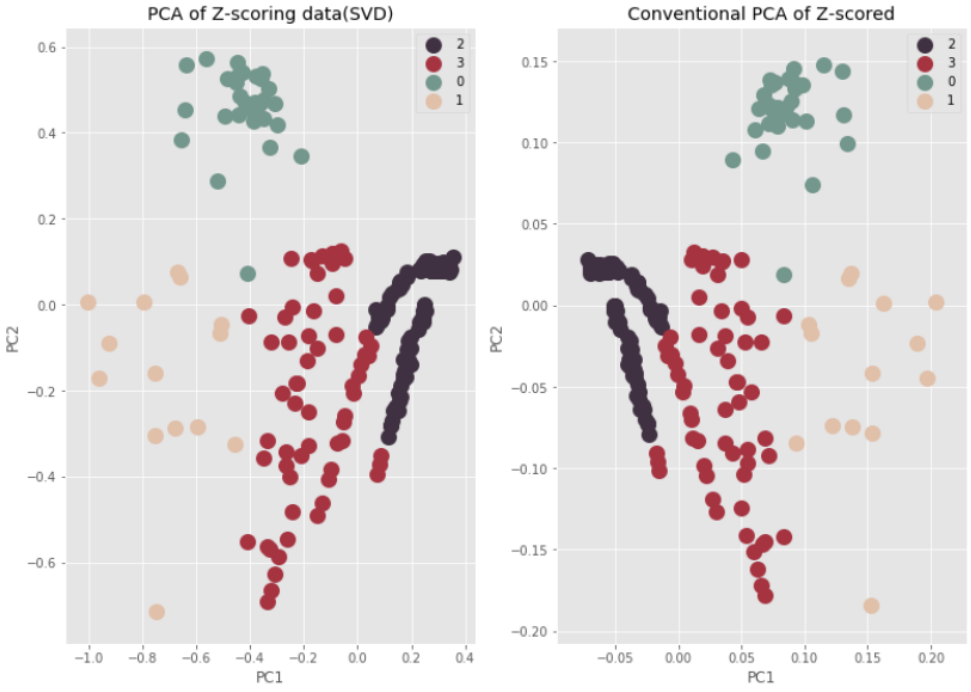

Fig. 12. PCA (left) and conventional PCA (right) of Z-scored standardized data

Graphs are mirrored and have different axes scales. Clusters are well separated in both graphs.

In the following graphs, you can see visualizations of PCAs of range standardized data and original dataset.

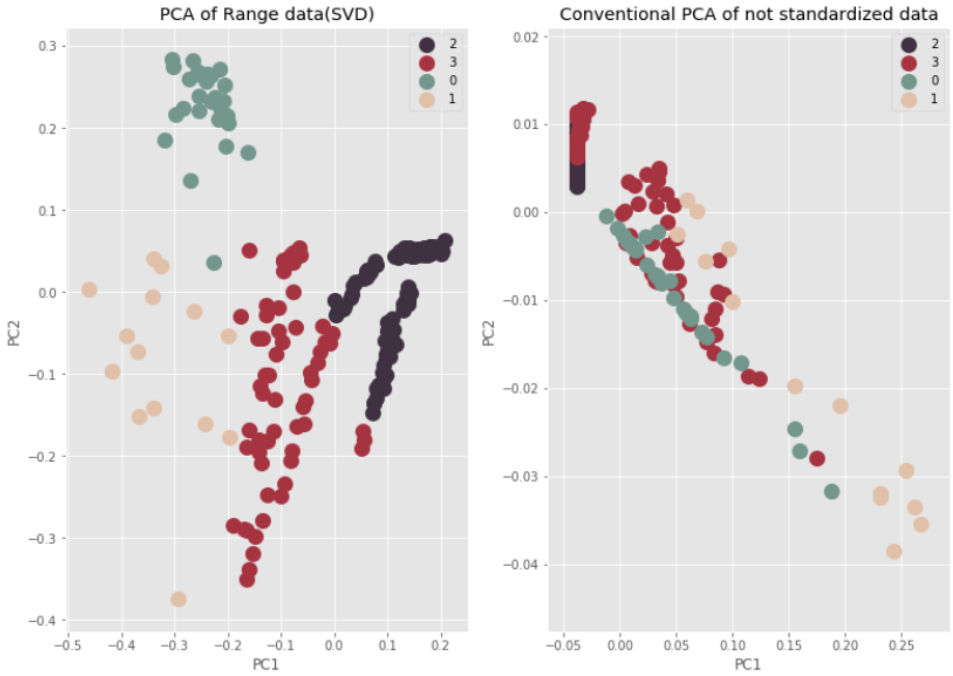

Fig. 13. PCAs of range standardized (left) and original (right) data

Visualization of PCA of range standardized data is quite similar with previous visualizations. Clusters are well separated.

Visualization of PCA of original data differs. Clusters are not separated; this visualization is not interpretable. This fact means that standardization is obligatory for clustering and other methods of data analysis.

In the following graph, you can see sklearn PCA visualization:

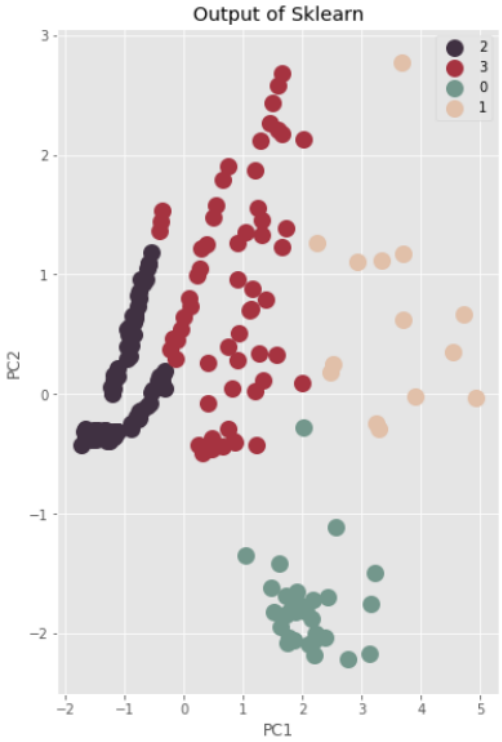

Fig. 14. sklearn PCA

The graph is mirrored visualization of PCA of range standardized data.

It is hard to say exactly which method of standardization is better, because they have very similar results. Z-score standardization can be easily interpreted. If the value equals 0, it means that the value is the mean. If the value is less or bigger than 0, it means that value differs from the mean, and absolute value shows how much and in what side it differs from the mean. In our point of view Z-scoring is better in the context of interpreting values in a dataset. But in fact, it doesn't influence on results greatly.

Correlation Coefficient

Before evaluating a correlation coefficients, 2 features of the "Star dataset" with the most similar to linear scatterplot were found. Scatterplots of all quantitative features ("Temperature", "Luminosity", "Radius", "Absolute magnitude") are shown on the graph below.

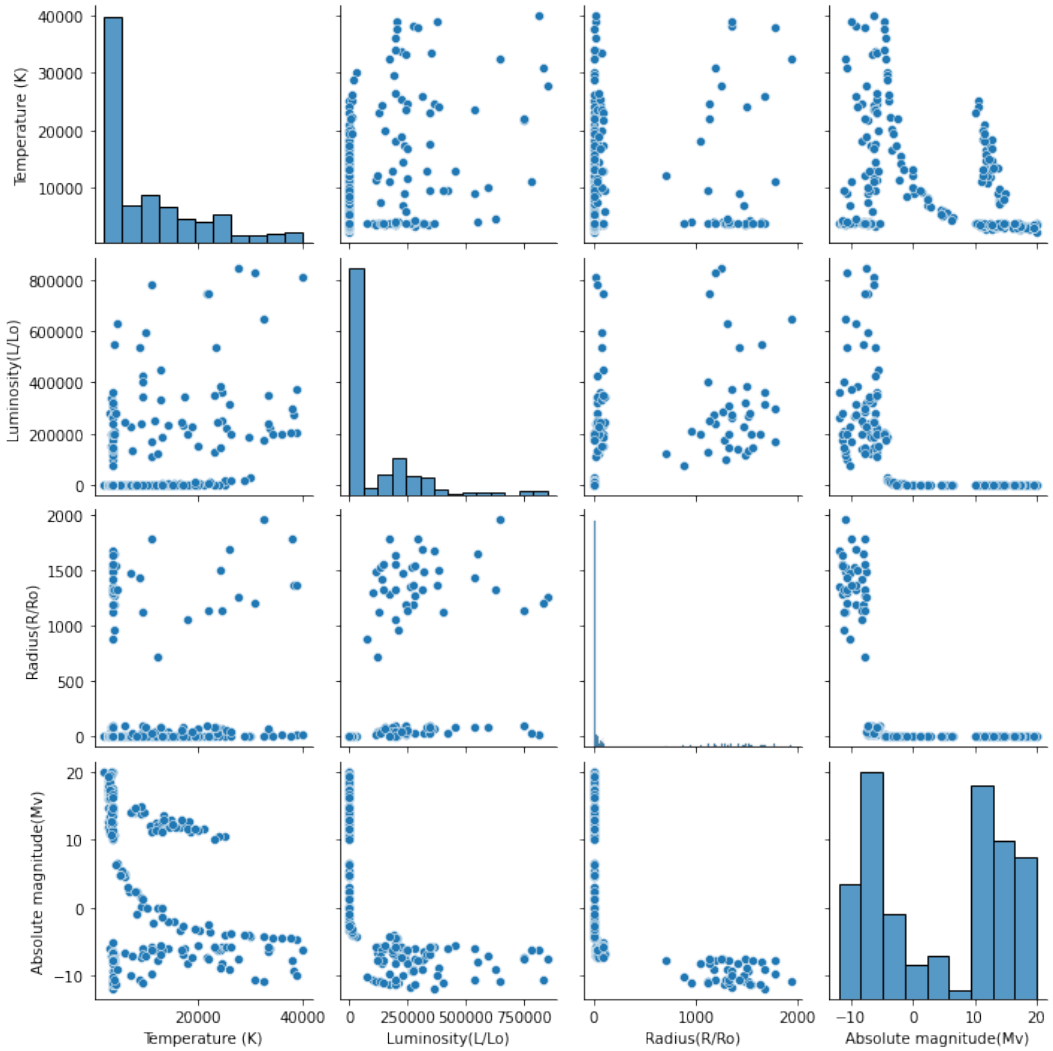

Fig. 15. Scatterplots of the quantitative features

According to the visual analysis of the graph, “Temperature” and “Absolute magnitude” have more linear-like scatterplot than other pairs. As for other pairs, it can be seen that there are relations between them, but they are far from linear. For almost every pair, there are clusters of points grouped around or along one value.

On the graph below the scatterplot for selected pair is shown.

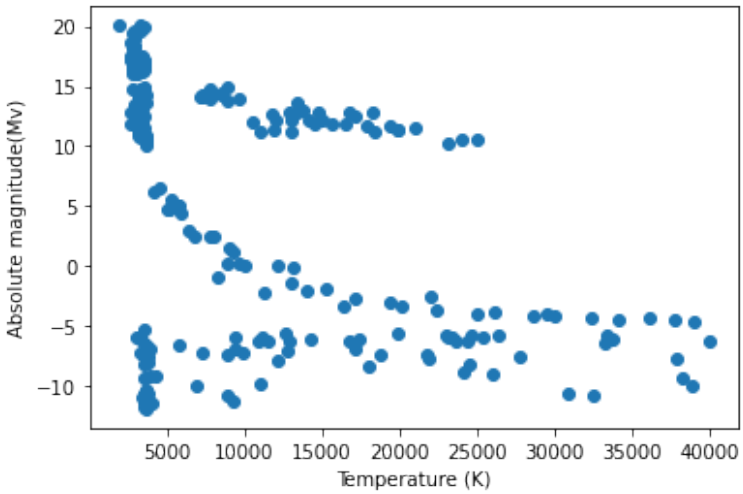

Fig. 16. Scatterplot for "Temperature" and "Absolute magnitude"

There is a linear trend called main sequence and two clusters above and below the line. Linear regression is suitable, since a slender linear-like dependency is visible. Presumably, the quality of the linear regression will not be high, as the main sequence looks a little hyperbolic on the left side, and two clusters above and below the line will not be reflected the linear model and will increase the error significantly.

Linear regression

After fitting of a linear regression, the following equation is obtained:

$$Y = -0.0004634\mathrm{X} + 9.247$$

where Y is an "Absolute magnitude" and X is a "Temperature".

On the following graph, there is a straight for this equation.

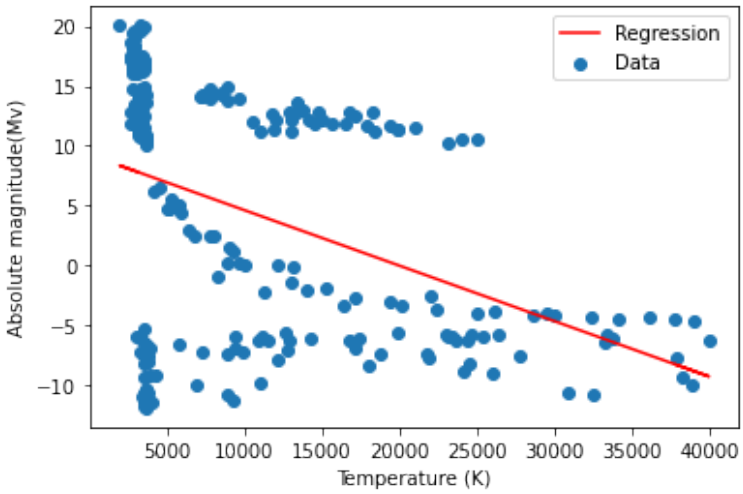

Fig. 17. Linear regression for "Temperature" and "Absolute magnitude"

It can be seen, that the slope is negative, as it should be for the main sequence of stars. It means that the absolute magnitude negatively depends on the temperature of the star. The greater the temperature, the lower the absolute magnitude.

Correlation coefficients

Correlation coefficients for all four features ("Temperature", "Luminosity", "Radius", "Absolute magnitude") were calculated and represented as a heatmap of the correlation matrix, which is on the following graph.

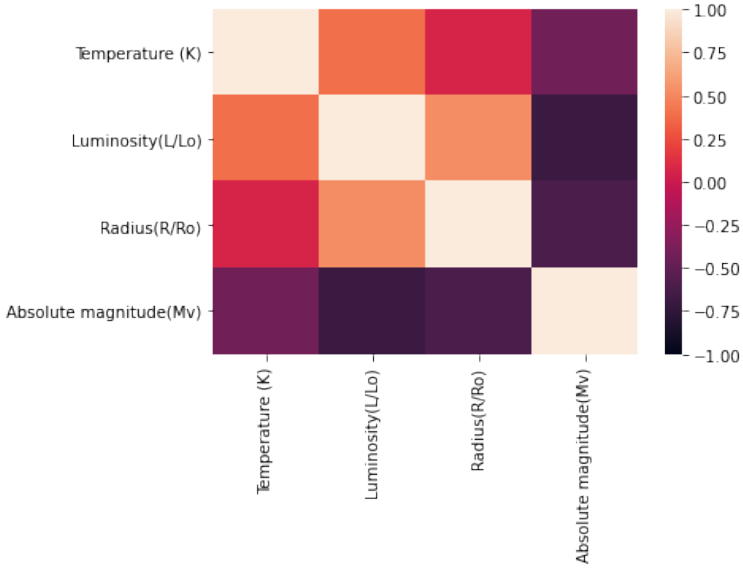

Fig. 18. Heatmap of correlation coefficients

Correlation coefficient for features “Temperature” and “Absolute magnitude” is -0.4203, which means that the correlation between them is quite weak and negative.

The strongest positive correlation is between “Luminosity” and “Radius”. The strongest negative correlation is between “Luminosity” and “Absolute magnitude”. The weakest correlation is observed for “Temperature” and “Radius”.

Determinacy coefficients

On the following graph the heatmap for determinacy coefficients matrix is presented.

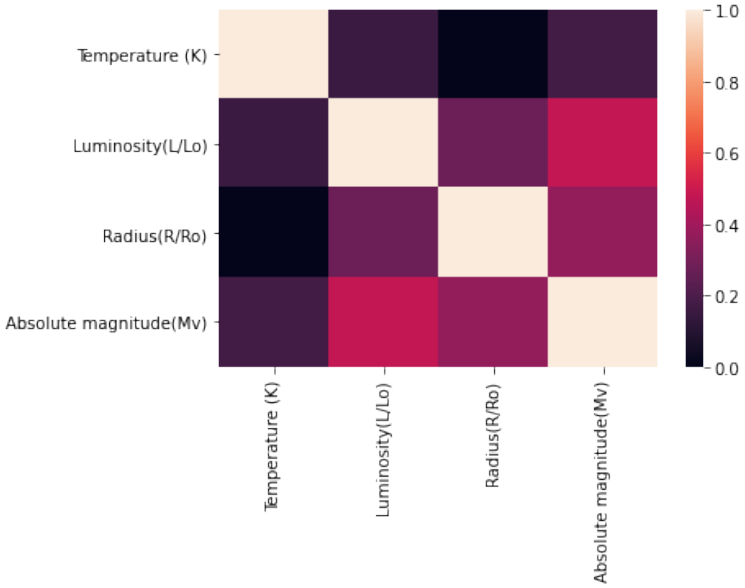

Fig. 19. Heatmap of determinacy coefficients

Determinacy coefficient for the features “Temperature” and “Absolute magnitude” is $R^2 = 0.1766$, which means that the variables explain the low proportion of variance of each other. This result is consistent with the presumption made before.

As known, the closer the value of the coefficient to 0, the weaker the dependence between the variables under consideration. In this case more than 80% of variance is unexplained because of two clusters of stars, stand out from the general pattern. The cluster above the line of the regression is the cluster of giants, the cluster above the line consists of white dwarfs. The stars in these clusters are characterized by extremely low dependence of absolute magnitude on temperature.

Upon the whole, “Absolute magnitude” and “Luminosity” have the highest determinacy coefficient, while “Temperature” and “Radius” have the lowest one.

Predictions

Let’s predict the absolute magnitude of the star by its temperature using estimated linear model.

We have three values of the temperature: 1 000, 25 000, 35000 Kelvins. After applying the linear equation, the result, presented on a graph, was obtained.

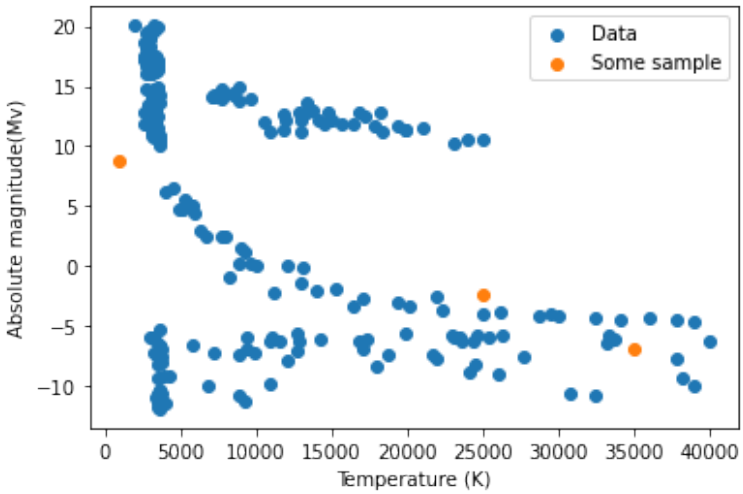

Fig. 20. Scatterplot with the predicted values

The model predicts the absolute magnitude by temperature quite well for the main sequence stars, but the prediction will always be wrong if the star is from another cluster (for example giants above the main sequence or white dwarfs below this line).

The mean relative absolute error MRAE of the model is 354.95 on the set of all points from the data. Such a big error is due to the fact that the data contains other types of stars in addition to the main sequence stars.

Appendix

K-Means

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

df = pd.read_csv('star_dataset.csv')
df.rename(columns={'Temperature (K)': 'Temperature', 'Luminosity(L/Lo)': 'Luminosity', 'Radius(R/Ro)':'Radius',
                   'Absolute magnitude(Mv)':'Absolute magnitude'}, inplace=True)
quantitative_columns = ['Temperature', 'Luminosity', 'Radius', 'Absolute magnitude']

X = df[quantitative_columns]


# Applying K-Means

from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

Ks = [4,7] #number of clusters

X_scaled = StandardScaler().fit_transform(X) #standardizing data

inertias = np.zeros((2,10)) #array, which contains inertias for each number of cluster (row)
                            #and for each initialization of KMeans (column)

best_inertias=[] #contains best_inertias for every cluster

j=0 #number of row in inertias
for k in Ks: #k is a number of clusters
    best_inertia = 1e10 #initial inertia

    for i in range(10):
        kmeans = KMeans(n_clusters=k, n_init = 1, max_iter = 500, init='random',
                            tol=1e-4, algorithm='full', random_state = i).fit(X_scaled)
                        #apply KMeans 10 times with k clusters, maximum number of iterations 500
                        #random initial cluster centers, relative tolerance 0.0001, EM-style algorythm
                        # and random state i

```

```
        inertias[j,i]=kmeans.inertia_

        tmp_labels = kmeans.labels_ #cluster labels for each object

        if inertias[j,i] < best_inertia: #choose the best clustering
            best_inertia = inertias[j,i] #save best_inertia
            best_labels = tmp_labels #save labels for best inertia
            best_random = i #save random_state for reproducing the results
    df['cluster_id_' + str(k)] = best_labels #sve cluster labels in additional column
    best_inertias = np.append(inertias, best_inertia)
    print("Best inertia for "+str(k)+"-cluster clustering:", best_inertia)
    print("Random state for this clustering: ", best_random)
    print()
    j+=1

df.to_csv("Stars_with_clusters.csv")
categorical_columns = ["Star type", "Star color", "Spectral Class"]
type_unique = df["Star type"].unique() #get unique values of star types
type_num = [len(df[df["Star type"]==el]) for el in type_unique] #number of elements for every type


plt.style.use('ggplot') #define the style of plotting

plt.figure(figsize=(15,7))
type_num_for_cluster = np.array([[len(df[(df.cluster_id_4==i)&(df["Star type"] == el)]) for el in type_unique]
                                 for i in range(4)]) #counting how many objects in each type belong to each cluster

# plotting bar graph with defined label and color
plt.bar(type_unique, type_num_for_cluster[0], label='0', color='#73978C')
plt.bar(type_unique, type_num_for_cluster[1],
bottom=type_num_for_cluster[0], label ='1', color='#E1C0A9')
plt.bar(type_unique, type_num_for_cluster[2],
bottom=type_num_for_cluster[1]+type_num_for_cluster[0], label ='2',
       color='#403142')
plt.bar(type_unique, type_num_for_cluster[3],
       bottom=type_num_for_cluster[2]+type_num_for_cluster[1]+type_num_for_cluster[0], label ='3', color='#A63340')

plt.legend() #define the legend
plt.xticks(type_unique, ['Red Dwarf', 'Brown Dwarf', 'White Dwarf', 'Main Sequence', 'SuperGiants', 'HyperGiants'],
           fontsize = 15, fontweight=400) #define the names of ticks (names of bars)

#define labels of axes
plt.ylabel("Star type")
plt.xlabel("Clusters")

```

```
plt.title("Star type clustering", fontsize = 20) #define title of the graph

#set limits of axes display
plt.gca().set_ylim(-1,43)
plt.gca().set_xlim(-1,6)

#show the graph
plt.show()


fig, ax = plt.subplots(figsize = (15,7)) #define the style of plotting

type_num_for_cluster = np.array([[len(df[(df.cluster_id_7==i)&(df["Star type"] == el)]) for el in type_unique]
                                 for i in range(7)]) #counting how many objects in each type belong to each cluster

# plotting bar graph with defined label and color
ax.bar(type_unique, type_num_for_cluster[0], label='0', color='#3a6170')
    ax.bar(type_unique, type_num_for_cluster[1], bottom=type_num_for_cluster[0], label ='1', color = '#73978C')
    ax.bar(type_unique, type_num_for_cluster[2], bottom=type_num_for_cluster[1]+type_num_for_cluster[0], label ='2',
           color='#E1C0A9')
    ax.bar(type_unique, type_num_for_cluster[3],
           bottom=type_num_for_cluster[2]+type_num_for_cluster[1]+type_num_for_cluster[0], label ='3', color='#403142')
    ax.bar(type_unique, type_num_for_cluster[4],
           bottom=type_num_for_cluster[3]+type_num_for_cluster[2]+type_num_for_cluster[1]+type_num_for_cluster[0],
           label ='4', color='#A63340')
    ax.bar(type_unique, type_num_for_cluster[5],
           bottom=type_num_for_cluster[4]+type_num_for_cluster[3]+type_num_for_cluster[2]+
           type_num_for_cluster[1]+type_num_for_cluster[0],label ='5', color='#EAE1D0')
    ax.bar(type_unique, type_num_for_cluster[6],
           bottom=type_num_for_cluster[5]+type_num_for_cluster[4]+type_num_for_cluster[3]+
           type_num_for_cluster[2]+type_num_for_cluster[1]+type_num_for_cluster[0], label ='6', color ='#ebbcc0')

    plt.legend() #define the legend

    plt.xticks(type_unique, ['Red Dwarf', 'Brown Dwarf', 'White Dwarf', 'Main Sequence', 'SuperGiants', 'HyperGiants'],
               fontsize = 15, fontweight=400) #define the names of ticks (names of bars)

    #define labels of axes
    plt.ylabel("Star type")
```

```
    plt.xlabel("Clusters")

    plt.title("Star type clustering", fontsize = 20) #define title of the
graph

    #set limits of axes display
    plt.gca().set_ylim(-1,43)
    plt.gca().set_xlim(-1,6)

    #show the graph
    plt.show()


    # Relative difference

    df_cluster = pd.read_csv("Stars_with_clusters.csv")

    relative_difference_4 = np.zeros((4,4))
    for j, feature in enumerate(quantitative_columns):
        grand_mean = np.mean(df_cluster[feature])
        for i in range(4):
            relative_difference_4[i,j] =
round(100*(np.mean(df_cluster[df_cluster["cluster_id_4"]==i][feature])/gra
nd_mean-1),2)

    relative_difference_7 = np.zeros((7,4))
    for j, feature in enumerate(quantitative_columns):
        grand_mean = np.mean(df_cluster[feature])
        for i in range(7):
            relative_difference_7[i,j] =
round(100*(np.mean(df_cluster[df_cluster["cluster_id_7"]==i][feature])/gra
nd_mean-1),2)


    #Quetelet indeces

    print(categorical_columns)
    types = df_cluster["Star type"].unique()
    print(types)
    print(len(types))

    colors = df_cluster["Star color"].unique()
    print(colors)
    print(len(colors))

    classes = df_cluster["Spectral Class"].unique()
    print(classes)
    print(len(classes))

    quetelet_indeces_types_4 = np.zeros((4,6))
    for j, cat in enumerate(types):
        for i in range(4):
```

```
            p_kv =
len(df_cluster[(df_cluster["cluster_id_4"]==i)&(df_cluster["Star
type"]==cat)])/240.
            p_k = len(df_cluster[df_cluster["cluster_id_4"]==i])/240.
            p_v = len(df_cluster[df_cluster["Star type"]==cat])/240.
            quetelet_indeces_types_4[i,j] = round(100*(p_kv/(p_k*p_v)-
1),2)

    quetelet_indeces_colors_4 = np.zeros((4,19))
    for j, cat in enumerate(colors):
        for i in range(4):
            p_kv =
len(df_cluster[(df_cluster["cluster_id_4"]==i)&(df_cluster["Star
color"]==cat)])/240
            p_k = len(df_cluster[df_cluster["cluster_id_4"]==i])/240.
            p_v = len(df_cluster[df_cluster["Star color"]==cat])/240.
            quetelet_indeces_colors_4[i,j] = round(100*(p_kv/(p_k*p_v)-
1),2)

    quetelet_indeces_classes_4 = np.zeros((4,7))
    for j, cat in enumerate(classes):
        for i in range(4):
            p_kv =
len(df_cluster[(df_cluster["cluster_id_4"]==i)&(df_cluster["Spectral
Class"]==cat)])/240.
            p_k = len(df_cluster[df_cluster["cluster_id_4"]==i])/240.
            p_v = len(df_cluster[df_cluster["Spectral Class"]==cat])/240.
            quetelet_indeces_classes_4[i,j] = round(100*(p_kv/(p_k*p_v)-
1),2)

    quetelet_indeces_types_7 = np.zeros((7,6))
    for j, cat in enumerate(types):
        for i in range(7):
            p_kv =
len(df_cluster[(df_cluster["cluster_id_7"]==i)&(df_cluster["Star
type"]==cat)])/240.
            p_k = len(df_cluster[df_cluster["cluster_id_7"]==i])/240.
            p_v = len(df_cluster[df_cluster["Star type"]==cat])/240.
            quetelet_indeces_types_7[i,j] = round(100*(p_kv/(p_k*p_v)-
1),2)

    quetelet_indeces_colors_7 = np.zeros((7,19))
    for j, cat in enumerate(colors):
        for i in range(7):
            p_kv =
len(df_cluster[(df_cluster["cluster_id_7"]==i)&(df_cluster["Star
color"]==cat)])/240
            p_k = len(df_cluster[df_cluster["cluster_id_7"]==i])/240.
            p_v = len(df_cluster[df_cluster["Star color"]==cat])/240.
            quetelet_indeces_colors_7[i,j] = round(100*(p_kv/(p_k*p_v)-
1),2)

    quetelet_indeces_classes_7 = np.zeros((7,7))
```

```
    for j, cat in enumerate(classes):
        for i in range(7):
            p_kv = len(df_cluster[(df_cluster["cluster_id_7"]==i)&(df_cluster["Spectral Class"]==cat)])/240.
            p_k = len(df_cluster[df_cluster["cluster_id_7"]==i])/240.
            p_v = len(df_cluster[df_cluster["Spectral Class"]==cat])/240.
            quetelet_indeces_classes_7[i,j] = round(100*(p_kv/(p_k*p_v)-1),2)
```

Bootstrap

```
import numpy as np
import pandas as pd

df = pd.read_csv('star_dataset.csv')
df.head()

n_cluster = 4
random_state = 9

from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

quantitative_columns = ['Temperature (K)', 'Luminosity(L/Lo)', 'Radius(R/Ro)', 'Absolute magnitude(Mv)']
X = df[quantitative_columns]
X = StandardScaler().fit_transform(X)

kmeans = KMeans(n_clusters=n_cluster, n_init=1, max_iter=500, init='random',
                tol=1e-4, algorithm='full', random_state=random_state)
kmeans.fit(X)
df['cluster_id'] = kmeans.labels_


def bootstrap(data, K):
    data = np.asarray(data)
    N = len(data)
    means = []
    for _ in range(K):
        idxs = np.random.choice(N, N, replace=True)
        mean = data[idxs].mean()
        means.append(mean)
    return np.asarray(means)

def confidence_interval(means, pivotal=True):
    if pivotal is True:
        left  = means.mean() - 1.96 * means.std()
        right = means.mean() + 1.96 * means.std()
    else:
```

```
        left  = np.percentile(means, 2.5)
        right = np.percentile(means, 97.5)
    return sorted([abs(left), abs(right)])

print('mean: {:.2f}'.format(df['Temperature (K)'].mean()))
print('no-pivotal: [{:.2f}, {:.2f}]'.format(*confidence_interval(bootstrap(df['Temperature (K)'], 1000), pivotal=False)))
print('pivotal: [{:.2f}, {:.2f}]'.format(*confidence_interval(bootstrap(df['Temperature (K)'], 1000), pivotal=True)))


cluster_0 = df[df['cluster_id'] == 0]
cluster_1 = df[df['cluster_id'] == 1]

data = bootstrap(cluster_0['Temperature (K)'], 1000) - bootstrap(cluster_1['Temperature (K)'], 1000)

print('no-pivotal: [{:.2f}, {:.2f}]'.format(*confidence_interval(data, pivotal=False)))
print('pivotal: [{:.2f}, {:.2f}]'.format(*confidence_interval(data, pivotal=True)))

cluster_0 = df
cluster_1 = df[df['cluster_id'] == 1]

data = bootstrap(cluster_0['Temperature (K)'], 1000) - bootstrap(cluster_1['Temperature (K)'], 1000)

print('no-pivotal: [{:.2f}, {:.2f}]'.format(*confidence_interval(data, pivotal=False)))
print('pivotal: [{:.2f}, {:.2f}]'.format(*confidence_interval(data, pivotal=True)))
```

Contigency table

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('star_dataset.csv')

# Numerical features
num_features = [
    'Temperature (K)', 'Luminosity(L/Lo)', 'Radius(R/Ro)', 'Absolute magnitude(Mv)'
]

fig, axs = plt.subplots(len(num_features), figsize=(8, 24))
for i, feature in enumerate(num_features):
    axs[i].hist(df[feature], int(np.round(np.sqrt(df.shape[0]))))
```

```
    axs[i].set_title('Histogram of %s feature' % feature)

temperature = df['Temperature (K)']
temperature_limit = temperature[temperature < 5000]

plt.hist(temperature_limit, int(np.round(np.sqrt(temperature_limit.shape[0
])))))
plt.title("Histogram of temperature feature")
plt.show()

def get_block(value, blocks_limit):
    for i, x in enumerate(blocks_limit):
        if value < x:
            return i
    return 0

temperature_blocks_limit = [3000, 3500, 4500, 10000, 15000, 40000]
df['Temperature_block'] = df['Temperature (K)'].apply(lambda x:
get_block(x, temperature_blocks_limit))

absolute_magnitude_blocks_limit = [-10, -5, 5, 12, 15, 20]
df['Absolute_magnitude_blocks'] = df['Absolute
magnitude(Mv)'].apply(lambda x: get_block(x,
absolute_magnitude_blocks_limit))

features = ['Star type', 'Temperature_block', 'Absolute_magnitude_blocks']

for feature in features:
    counts = df[feature].value_counts()
    print(counts)
    print()

_TOTAL = 'total'

def q_index(cond_row):
    _total = cond_row[_TOTAL]
    return cond_row.apply(
        lambda x: (x-_total)/_total if x != _total else _total
    )

type_temperature = pd.crosstab(
    df['Temperature_block'],
    df['Star type'],
    margins = True,
    margins_name=_TOTAL,
    rownames=['Temperature'],
    colnames=['Star type']
)
type_temperature

type_temperature_prob = pd.crosstab(
    df['Temperature_block'],
    df['Star type'],
```

```
    margins = True,
    margins_name='_TOTAL',
    rownames=['Temperature Prob'],
    colnames=['Star type'],
    normalize='all'
)
type_temperature_prob

type_magnitude = pd.crosstab(
    df['Absolute_magnitude_blocks'],
    df['Star type'],
    margins = True,
    margins_name='_TOTAL',
    rownames=['Absolute magnitude'],
    colnames=['Star type']
)
type_magnitude

type_magnitude_prob = pd.crosstab(
    df['Absolute_magnitude_blocks'],
    df['Star type'],
    margins = True,
    margins_name='_TOTAL',
    rownames=['Absolute magnitude'],
    colnames=['Star type'],
    normalize='all'
)
type_magnitude_prob

type_temperature_cond = pd.crosstab(
    df['Temperature_block'],
    df['Star type'],
    margins = True,
    margins_name='_TOTAL',
    rownames=['Temperature_cond'],
    colnames=['Star type'],
        normalize='columns'
    )
    type_temperature_cond

    type_magnitude_cond = pd.crosstab(
        df['Absolute_magnitude_blocks'],
        df['Star type'],
        margins = True,
        margins_name='_TOTAL',
        rownames=['Absolute magnitude'],
        colnames=['Star type'],
        normalize='columns'
    )
    type_magnitude_cond

    type_temperature_quetelet = type_temperature_cond.apply(q_index,
axis=1)
```

```
    type_temperature_quetelet

    type_magnitude_quetelet = type_magnitude_cond.apply(q_index, axis=1)
    type_magnitude_quetelet

    def chi2(freq_crosstab, prob_crosstab):
        return freq_crosstab.combine(
                prob_crosstab,
                lambda a, b: (b - a) ** 2 / a
        ).drop([_TOTAL], axis=0) \
         .drop([_TOTAL], axis=1) \
         .values.sum()

    def quetelet_summary(prob_crosstab, q_crosstab):
        return prob_crosstab \
                .combine(q_crosstab, np.multiply) \
                .drop([_TOTAL], axis=0) \
                .drop([_TOTAL], axis=1) \
                .values.sum()

    def freq_crosstab(prob_crosstab):
        return prob_crosstab.apply(lambda r: r[_TOTAL] * dd[_TOTAL]) \
        .drop([_TOTAL], axis=0) \
        .drop([_TOTAL], axis=1)

    type_temperature_freq = freq_crosstab(type_temperature_prob)
    type_temperature_freq

    type_magnitude_freq = freq_crosstab(type_magnitude_prob)
    type_magnitude_freq

    q = quetelet_summary(type_temperature_prob,
type_temperature_quetelet)
    print('Average Quetelet index:', q)
    print('Chi2:', chi2(type_temperature_freq, type_temperature_prob))

    q = quetelet_summary(type_magnitude_prob, type_magnitude_quetelet)
    print('Average Quetelet index:', q)
    print('Chi2:', chi2(type_magnitude_freq, type_magnitude_prob))

    def plot_heat_map(prob, quetelet):
        heat_map_quet_T_G = (prob * quetelet) \
        .drop([_TOTAL], axis=0) \
        .drop([_TOTAL], axis=1) \
        .to_numpy()

        plt.imshow(
            heat_map_quet_T_G,
            cmap='hot',
            interpolation='nearest'
        )
        plt.colorbar()
        plt.show()
```

```
168.
169.     plot_heat_map(type_temperature_prob, type_temperature_quetelet)
170.     plot_heat_map(type_magnitude_prob, type_magnitude_quetelet)
```

PCA SVD

```
1.import numpy as np
2.import pandas as pd
3.import matplotlib.pyplot as plt
4.import numpy.linalg as lg
5.import seaborn as sbn
6.from sklearn.linear_model import LinearRegression
7.%matplotlib inline
8.
9.df = pd.read_csv('star_dataset.csv')
10.
11.def standardizer(x):
12.
13.    """
14.        standardize entity-to-feature data matrix by
15.          applying Z-scoring, Range standardization and Rank methods
16.
17.        Arguments:
18.            x, numpy array, entity-to-feature data matrix
19.        Returns:
20.            Z-scored and Range standardized data matrices, x_zscore,
   x_range, x_rank.
21.    """
22.
23.    x_ave = np.mean(x, axis=0)
24.    x_min = np.min(x, axis=0)
25.    x_rng = np.max(x, axis=0)-np.min(x, axis=0)
26.    x_std = np.std(x, axis=0)
27.    x_zscore = np.divide(np.subtract(x, x_ave), x_std)    # Z-scoring
   standardization/ StandardScaler
28.    x_range = np.divide(np.subtract(x, x_ave), x_rng)  # Range
   standardization
29.    x_rank = np.divide(np.subtract(x, x_min), x_rng) # Rank
   standardization
30.    return x_zscore, x_range, x_rank
31.
32.def singular_decomposition(x):
33.    z, mu, c = lg.svd(x, full_matrices=True)
34.    z = -z
35.    c = -c
36.    mu_arg_max = np.argmax(mu)
37.
38.    n_contributions = np.power(mu, 2)
39.    print("n_contributions: ", n_contributions)
40.
41.    ds = np.sum(np.power(mu, 2))  # Data Scatter
42.    print("ds: ", ds)
```

```
43.
44.     # Determine contributions of all the principal components to the data
    scatter
45.     p_contributions = np.divide(n_contributions, ds)
46.
47.     return z, mu, c, ds, n_contributions, p_contributions, mu_arg_max
48.
49.df.rename(columns={'Temperature
    (K)': 'Temperature', 'Luminosity(L/Lo)': 'Luminosity', 'Radius(R/Ro)':'Rad
    ius',
50.                  'Absolute magnitude(Mv)':'Absolute magnitude'},
    inplace=True)
51.quantitative_columns = ['Temperature', 'Luminosity', 'Radius', 'Absolute
    magnitude']
52.
53.X = df[quantitative_columns]
54.
55.x_zscore, x_range, x_rank = standardizer(x=X)
56.
57.# Singular decomposition on original data
58.z_x, mu_x, c_x, ds_x, n_contributions_x, p_contributions_x, mu_argmax_x =
    \
59.singular_decomposition(x=X)
60.print("SVD")
61.print("U:", z_x)
62.print("Sigma:", mu_x)
63.print("V*:", c_x)
64.
65.# Singular decomposition on Z-Scored standardized data
66.z_x_zscore, mu_x_zscore, c_x_zscore, ds_x_zscore,\
67.n_contributions_x_zscore, p_contributions_x_zscore, mu_argmax_x_zscore  =
    \
68.singular_decomposition(x=x_zscore)
69.print("SVD")
70.print("U:", z_x_zscore)
71.print("Sigma:", mu_x_zscore)
72.print("V*:", c_x_zscore)
73.
74.# Singular decomposition on Range standardized data
75.z_x_range, mu_x_range, c_x_range, ds_x_range,\
76.n_contributions_x_range, p_contributions_x_range,mu_argmax_x_range = \
77.singular_decomposition(x=x_range)
78.print("SVD")
79.print("U:", z_x_range)
80.print("Sigma:", mu_x_range)
81.print("V*:", c_x_range)
82.
83.# Singular decomposition on Rank standardized data
84.z_x_rank, mu_x_rank, c_x_rank, ds_x_rank, \
85.n_contributions_x_rank, p_contributions_x_rank, mu_argmax_x_rank = \
86.singular_decomposition(x=x_rank)
87.print("SVD")
88.print("U:", z_x_rank)
```

```
print("Sigma:", mu_x_rank)
print("V*:", c_x_rank)

print("Natural contributions of \n",
      "Original data       :", n_contributions_x, "\n",
      "Ranked std data     :", n_contributions_x_rank, "\n",
      "Ranged std data     :", n_contributions_x_range, "\n",
      "Z-Scored std data   :", n_contributions_x_zscore,)

print("Percent contributions of \n",
      "Original data       :\n", 100*p_contributions_x, "\n",
          "Ranked std data     :\n", 100*p_contributions_x_rank, "\n",
          "Ranged std data     :\n", 100*p_contributions_x_range, "\n",
          "Z-Scored std data   :\n", 100*p_contributions_x_zscore,)

    print("Data scetter \n",
          "Original data       :", ds_x, "\n",
          "Rank std data     :", ds_x_rank, "\n",
          "Range std data     :", ds_x_range, "\n",
          "Z-Score std data   :", ds_x_zscore,)

    def part_3(x, z, c, mu, argmax):
        z1 = z[:, argmax]  # hiden score
        c1 = c[argmax, :]  # loading
        mu1 = mu[argmax]
        print("c1             :", c1)
        max_val = x.max()
        alpha = 100/np.sum(c1*max_val)
        loading = c1*alpha
        print("Loading        :", loading)
        ranking_factors = x.dot(c1)
        rank_min = min(ranking_factors)
        rank_max = max(ranking_factors)
        ranking_factors = (ranking_factors - rank_min)/(rank_max -
rank_min)
        arg_top5 = np.argsort(ranking_factors)[::-1][:5]
        top_5 = np.sort(ranking_factors,)[::-1][:5]

        contribution = 100*mu[0]**2/((x*x).sum().sum())
        print("Contribution: ", contribution)

        print("Top5 stars:", top_5)
        return loading, ranking_factors, arg_top5

    _, ranking_factors, arg_top5 = part_3(x=X, z=z_x, c=c_x, mu=mu_x,
argmax=mu_argmax_x)

    df.loc[arg_top5]

    _, ranking_factors, arg_top5 = part_3(x=x_zscore, z=z_x_zscore,
c=c_x_zscore, mu=mu_x_zscore, argmax=mu_argmax_x_zscore)

    df.loc[arg_top5]
```

```
139.
140.     _, ranking_factors, arg_top5 = part_3(x=x_range, z=z_x_range,
    c=c_x_range, mu=mu_x_range, argmax=mu_argmax_x_range)
141.
142.     df.loc[arg_top5]
143.
144.     _, _, arg_top5 =\
145.     part_3(x=x_rank, z=z_x_rank, c=c_x_rank, mu=mu_x_rank,
    argmax=mu_argmax_x_rank)
146.
147.     df.loc[arg_top5]
148.
149.     df_labeled = pd.read_csv("Stars_with_clusters.csv")
150.
151.     group_0 = np.where(df_labeled["cluster_id_4"] == 0)[0]
152.     print(group_0)
153.
154.     group_1 = np.where(df_labeled["cluster_id_4"] == 1)[0]
155.     print(group_1)
156.
157.     group_2 = np.where(df_labeled["cluster_id_4"] == 2)[0]
158.     print(group_2)
159.
160.     group_3 = np.where(df_labeled["cluster_id_4"] == 3)[0]
161.     print(group_3)
162.
163.     len(group_1)+len(group_2)+len(group_3)+len(group_0)
164.
165.     z_z0 = z_x_zscore[:, 0]*np.sqrt(mu_x_zscore[0])
166.     z_z1 = z_x_zscore[:, 1]*np.sqrt(mu_x_zscore[1])
167.
168.     z_r0 = z_x_range[:, 0]*np.sqrt(mu_x_range[0])
169.     z_r1 = z_x_range[:, 1]*np.sqrt(mu_x_range[1])
170.
171.     plt.style.use('ggplot')
172.
173.     fig = plt.figure(figsize=(15, 9.5))
174.
175.     ax = fig.add_subplot(121)
176.     ax.scatter(z_z0[group_2], z_z1[group_2], c='#403142', linewidths = 7,
    label ='2')
177.     ax.scatter(z_z0[group_3], z_z1[group_3], c='#A63340', linewidths = 7,
    label ='3')
178.     ax.scatter(z_z0[group_0], z_z1[group_0], c='#73978C', linewidths = 7,
    label='0')
179.     ax.scatter(z_z0[group_1], z_z1[group_1], c='#E1C0A9', linewidths = 7,
    label ='1')
180.     plt.title("PCA of Z-scoring standardized data", fontsize = 20)
181.     plt.legend()
182.     plt.xlabel("PC1")
183.     plt.ylabel("PC2")
184.     plt.xticks(fontsize = 13)
185.     plt.yticks(fontsize = 13)
```

```
186.
187.     ax = fig.add_subplot(122)
188.     ax.scatter(z_r0, z_r1)
189.     ax.scatter(z_r0[group_2], z_r1[group_2], c='#403142', linewidths = 7,
    label ='2')
190.     ax.scatter(z_r0[group_3], z_r1[group_3], c='#A63340', linewidths = 7,
    label ='3')
191.     ax.scatter(z_r0[group_0], z_r1[group_0], c='#73978C', linewidths = 7,
    label='0')
192.     ax.scatter(z_r0[group_1], z_r1[group_1], c='#E1C0A9', linewidths = 7,
    label ='1')
193.
194.     plt.legend()
195.
196.     plt.title("PCA of Range standardatized data", fontsize = 20)
197.     plt.xlabel("PC1")
198.     plt.ylabel("PC2")
199.     plt.xticks(fontsize = 13)
200.     plt.yticks(fontsize = 13)
201.     plt.show()
202.
203.     # Loadings:
204.     c_x_zscore[:3, :]
205.
206.     def conventional_pca(X, standardized=False):
207.         # Let us name x as original data set,
208.         # restricted at selected subset of features, i.e. x_selected.
209.         if standardized is False:
210.             mean_x = np.mean(X, axis=0)
211.             Y = np.subtract(X, mean_x)  # centered version
212.             B = (Y.T@Y)/Y.shape[0]  # covariance matrix of Y
213.             L, C = np.linalg.eig(B)  # Eigenvalues
214.             sorted_idx = np.argsort(L)[::-1]  # descending order
215.             la1 = L[sorted_idx[0]]
216.             c1 = C[:, sorted_idx[0]]  # unlike np.linalg.svd now we
    should consider the column
217.             pc1 = np.divide(Y@c1, np.sqrt(Y.shape[0]*la1))  # 1st
    principle component
218.             B_dot = B - la1*np.multiply(c1, c1.T)  # Residual of Cov.
219.             L_, C_ = np.linalg.eig(B_dot)
220.             argmax_ = np.argmax(L_)
221.             la2 = L_[argmax_]
222.             c2 = C_[:, argmax_]
223.             pc2 = np.divide(Y@c2, np.sqrt(Y.shape[0]*la2))  # 2nd
    principle component
224.         else:
225.             Y = X
226.             B = (Y.T@Y)/Y.shape[0]  # covariance matrix of Y
227.             L, C = np.linalg.eig(B)
228.             sorted_idx = np.argsort(L)[::-1]  # descending order
229.             la1 = L[sorted_idx[0]]
230.             c1 = C[:, sorted_idx[0]]  # unlike np.linalg.svd now we
    should consider the column
```

```
            pc1 = np.divide(Y@c1, np.sqrt(Y.shape[0]*la1))  # 1st
principle component
            la2 = L[sorted_idx[1]]
            c2 = -C[:, sorted_idx[1]]  # unlike np.linalg.svd now we
should consider the column
            pc2 = np.divide(Y@c2, np.sqrt(Y.shape[0]*la2))  # 2nd
principle component

        return pc1, pc2

    pc1_x, pc2_x = conventional_pca(X=X, standardized=False)

    pc1_x_z, pc2_x_z = conventional_pca(X=x_zscore, standardized=True)

    from sklearn.decomposition import PCA
    pca = PCA(n_components=2)
    pcas = pca.fit_transform(x_zscore)
    pc1, pc2 = pcas[:, 0], pcas[:, 1]

    fig = plt.figure(figsize=(20, 20))

    ax = fig.add_subplot(231)
    ax.scatter(z_z0, z_z1)
    ax.scatter(z_z0[group_2], z_z1[group_2], c='#403142', linewidths = 7,
label ='2')
    ax.scatter(z_z0[group_3], z_z1[group_3], c='#A63340', linewidths = 7,
label ='3')
    ax.scatter(z_z0[group_0], z_z1[group_0], c='#73978C', linewidths = 7,
label='0')
    ax.scatter(z_z0[group_1], z_z1[group_1], c='#E1C0A9', linewidths = 7,
label ='1')
    plt.title("PCA of Z-scoring data(SVD)")
    plt.xlabel("PC1")
    plt.ylabel("PC2")
    plt.legend()

    ax = fig.add_subplot(232)
    ax.scatter(pc1_x_z, pc2_x_z)
    ax.scatter(pc1_x_z[group_2], pc2_x_z[group_2], c='#403142',
linewidths = 7, label ='2')
    ax.scatter(pc1_x_z[group_3], pc2_x_z[group_3], c='#A63340',
linewidths = 7, label ='3')
    ax.scatter(pc1_x_z[group_0], pc2_x_z[group_0], c='#73978C',
linewidths = 7, label='0')
    ax.scatter(pc1_x_z[group_1], pc2_x_z[group_1], c='#E1C0A9',
linewidths = 7, label ='1')
    plt.title("Conventional PCA of Z-scored")
    plt.xlabel("PC1")
    plt.ylabel("PC2")
    plt.legend()

    ax = fig.add_subplot(233)
    ax.scatter(pc1, pc2)
```

```
    ax.scatter(pc1[group_2], pc2[group_2], c='#403142', linewidths = 7, label ='2')
    ax.scatter(pc1[group_3], pc2[group_3], c='#A63340', linewidths = 7, label ='3')
    ax.scatter(pc1[group_0], pc2[group_0], c='#73978C', linewidths = 7, label='0')
    ax.scatter(pc1[group_1], pc2[group_1], c='#E1C0A9', linewidths = 7, label ='1')
    plt.title("Output of Sklearn")
    plt.xlabel("PC1")
    plt.ylabel("PC2")
    plt.legend()

    ax = fig.add_subplot(234)
    plt.scatter(z_r0, z_r1)
    ax.scatter(z_r0[group_2], z_r1[group_2], c='#403142', linewidths = 7, label ='2')
    ax.scatter(z_r0[group_3], z_r1[group_3], c='#A63340', linewidths = 7, label ='3')
    ax.scatter(z_r0[group_0], z_r1[group_0], c='#73978C', linewidths = 7, label='0')
    ax.scatter(z_r0[group_1], z_r1[group_1], c='#E1C0A9', linewidths = 7, label ='1')
    plt.title("PCA of Range data(SVD)")
    plt.xlabel("PC1")
    plt.ylabel("PC2")
    plt.legend()

    ax = fig.add_subplot(235)
    ax.scatter(pc1_x, pc2_x)
    ax.scatter(pc1_x[group_2], pc2_x[group_2], c='#403142', linewidths = 7, label ='2')
    ax.scatter(pc1_x[group_3], pc2_x[group_3], c='#A63340', linewidths = 7, label ='3')
    ax.scatter(pc1_x[group_0], pc2_x[group_0], c='#73978C', linewidths = 7, label='0')
    ax.scatter(pc1_x[group_1], pc2_x[group_1], c='#E1C0A9', linewidths = 7, label ='1')
    plt.title("Conventional PCA of not standardized data")
    plt.xlabel("PC1")
    plt.ylabel("PC2")
    plt.legend()

    plt.show()
```

Correlation coefficient

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

```

```
6.
7.df = pd.read_csv('star_dataset.csv')
8.df.head()
9.
10.sns.pairplot(data=df[['Temperature
   (K)', 'Luminosity(L/Lo)', 'Radius(R/Ro)', 'Absolute magnitude(Mv)']])
11.
12.plt.scatter(df['Temperature (K)'], df['Absolute magnitude(Mv)'])
13.plt.xlabel('Temperature (K)')
14.plt.ylabel('Absolute magnitude(Mv)')
15.plt.show()
16.
17.
18.from sklearn import linear_model
19.
20.regr = linear_model.LinearRegression()
21.regr.fit(df[['Temperature (K)']], df['Absolute magnitude(Mv)'])
22.print("y = {:.4}x + {:.4}".format(regr.coef_[0], regr.intercept_))
23.
24.plt.plot(df[['Temperature (K)']], regr.predict(df[['Temperature (K)']]),
   color='r', label="Regression")
25.plt.scatter(df['Temperature (K)'], df['Absolute magnitude(Mv)'],
   label='Data')
26.plt.xlabel('Temperature (K)')
27.plt.ylabel('Absolute magnitude(Mv)')
28.plt.legend()
29.plt.show()
30.
31.corr = df[['Temperature
   (K)', 'Luminosity(L/Lo)', 'Radius(R/Ro)', 'Absolute
   magnitude(Mv)']].corr()
32.sns.heatmap(corr,
33.        xticklabels=corr.columns,
34.        yticklabels=corr.columns, vmin=-1, vmax=1);
35.
36.print("Correlation coefficients for (Temperature (K), Absolute
   magnitude(Mv)): {:.4}".
37.      format((corr['Temperature (K)']['Absolute magnitude(Mv)'])))
38.
39.sns.heatmap(corr ** 2,
40.        xticklabels=corr.columns,
41.        yticklabels=corr.columns, vmin=0, vmax=1);
42.
43.print("Determinacy coefficients for (Temperature (K), Absolute
   magnitude(Mv)): {:.4}".
44.      format((corr['Temperature (K)']['Absolute magnitude(Mv)'] ** 2)))
45.
46.some_temperatures = np.array([1000, 25000, 35000])
47.
48.plt.scatter(df['Temperature (K)'], df['Absolute magnitude(Mv)'],
   label='Data')
49.plt.scatter(some_temperatures, regr.predict(some_temperatures.reshape(-
   1, 1)), label='Some sample')
```

```
plt.xlabel('Temperature (K)')
plt.ylabel('Absolute magnitude(Mv)')
plt.legend()
plt.show()

from sklearn.metrics import mean_absolute_error
y_true = df['Absolute magnitude(Mv)']
y_pred = regr.predict(df[['Temperature (K)']])

mrae = 100 * np.mean(np.absolute(np.divide(y_true - y_pred, y_true)))

print("MRAE {:.5}".format(mrae))
```