Задача 5

Необходимо провести анализ датасета (из задания 6) и сделать обработку данных по предложенному алгоритму. Код подготовить в виде файлов \*.py и сделать отчет в виде ноутбука с описанием процесса анализа.

Таблица 1. Поставленная задача.

|  |  |
| --- | --- |
| Номер темы | Основная тема |
| 10 | Кластеризация или классификация экзопланет |

1. Сколько в датасете объектов и признаков? Дать описание каждому признаку, если оно есть.

В датасете имеется 3584 наблюдений и 25 признаков:

1. PlanetIdentifier - уникальное название планеты;
2. TypeFlag - тип планеты (одиночная планета, жизнь у двойных звезд - одинарная или двойная орбита);
3. PlanetaryMassJpt - Масса планеты относительно Юпитера;
4. RadiusJpt - Радиус планеты относительно юпитера;
5. PeriodDays - Орбитальный период;
6. SemiMajorAxisAU - большая полуось относительно родительской планеты или двух планет;
7. Eccentricity - эксцентриситет планеты;
8. PeriastronDeg - долгота периапсиса;
9. LongitudeDeg - Средняя долгота;
10. AscendingNodeDeg - Долгота восходящего узла;
11. InclinationDeg - Наклон долготы;
12. SurfaceTempK - Температура поверхности в Кельвинах;
13. AgeGyr - Возраст планеты;
14. DiscoveryMethod - Метод обнаружения планеты;
15. DiscoveryYear - Год обнаружения планеты;
16. LastUpdated - Дата обновления информации о планете;
17. RightAscension - Прямое восхождение;
18. Declination - Склонение планеты;
19. DistFromSunParsec - расстояние до солнца в Парсеках;
20. HostStarMassSlrMass - масса звезды, вокруг которой крутится планета;
21. HostStarRadiusSlrRad - радиус звезды, вокруг которой крутится планета;
22. HostStarMetallicity - металличность звезды, вокруг которой крутится планета;
23. HostStarTempK - температура поверхности звезды, вокруг которой крутится планета;
24. HostStarAgeGyr - возраст звезды, вокруг которой крутится планета;
25. ListsPlanetIsOn - Категория планеты - подтвержденная, сомнительная или KOI.
26. Сколько категориальных признаков, какие?

Всего в датасете наблюдается 3 категориальных признака, а именно:

1. TypeFlag - тип планеты;
2. DiscoveryMethod - Метод обнаружения планеты;
3. ListsPlanetIsOn - Категория планеты.
4. Столбец с максимальным количеством уникальных значений категориального признака?

У ListsPlanetIsOn имеется 13 уникальных значений. Наиболее часто встречающиеся - Confirmed planets, то есть подтвержденные планеты.

1. Есть ли бинарные признаки?

Таковых не наблюдается. AgeGyr не может являться таковым, так как на вопрос "Какой у планеты возраст?" ответ "да" или "нет" не имеет смысла, а у самого столбца отсутствует 99,9% данных.

1. Какие числовые признаки?

Всего имеется 19 числовых признаков:

1. PlanetaryMassJpt - Масса планеты относительно Юпитера;
2. RadiusJpt - Радиус планеты относительно юпитера;
3. PeriodDays - Орбитальный период;
4. SemiMajorAxisAU - большая полуось относительно родительской планеты или двух планет;
5. Eccentricity - эксцентриситет планеты;
6. PeriastronDeg - долгота периапсиса;
7. LongitudeDeg - Средняя долгота;
8. AscendingNodeDeg - Долгота восходящего узла;
9. InclinationDeg - Наклон долготы;
10. SurfaceTempK - Температура поверхности в Кельвинах;
11. AgeGyr - Возраст планеты;
12. RightAscension - Прямое восхождение;
13. Declination - Склоение планеты;
14. DistFromSunParsec - расстояние до солнца в Парсеках;
15. HostStarMassSlrMass - масса звезды, вокруг которой крутится планета;
16. HostStarRadiusSlrRad - радиус звезды, вокруг которой крутится планета;
17. HostStarMetallicity - металличность звезды, вокруг которой крутится планета;
18. HostStarTempK - температура поверхности звезды, вокруг которой крутится планета;
19. HostStarAgeGyr - возраст звезды, вокруг которой крутится планета.
20. Есть ли пропуски?

Наблюдаются пропуски, в некоторых столбцах более 90% значений - пропуски. Сразу же можно увидеть столбцы, которые определенно требуют обработки или удаления. Так, столбец PlanetaryMassJpt на 63% из отсутствующих данных. Вряд-ли данные возможно "примерно" восстановить с помощью средних значений, поэтому этот столбец следует удалить. Удалению также подлежат SemiMajorAxisAU с 60% отсутствующих данных, Eccentricity с 70%, PeriastronDeg с 90%, LongitudeDeg и AscendingNodeDeg с 99%, InclinationDeg c 81%, SurfaceTempK с 80%, AgeGyr с 100%, HostStarAgeGyr с 85%.

1. Сколько объектов с пропусками?

3584 объектов (то есть 100%) имеют пропуски.

1. Столбец с максимальным количеством пропусков?

Больше всего пропусков в AgeGyr - из 3584 наблюдений лишь 2 имеют значение.

1. Есть ли на ваш взгляд выбросы, аномальные значения?

Для нахождения выбросов и аномальных значений воспользуемся ящиком с усами.

Так, ящик с усами для массы планеты представлен на Рисунке 1:

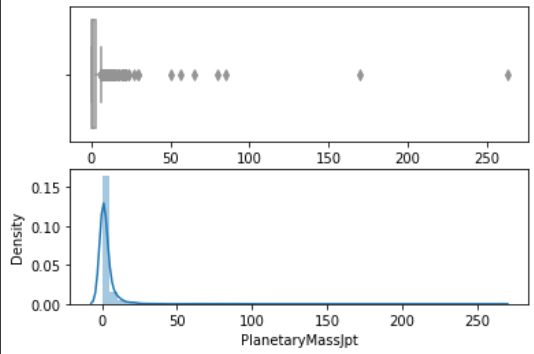


Рисунок . Ящик с усами для массы планет.

Планеты, выходящие за ящик с усами, действительно являются аномалиями, но из-за того, что некоторые значения действительно большие, их стоит сократить.

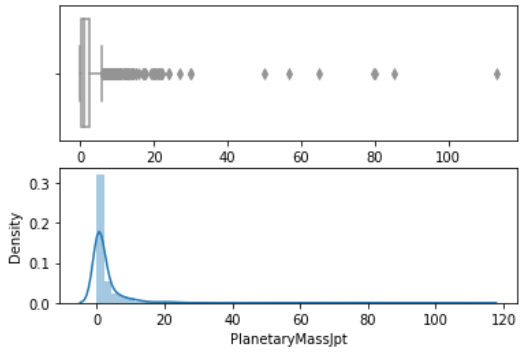


Рисунок 2. Ящик с усами для массы планет после изменения некоторых аномальных значений.

Продолжим использовать ящик с усами на всех оставшихся столбцов с действительными числами, пока не будут обработаны все данные. В итоге лишь у двух столбцов не оказалось выбросов – у средней долготы и долготы восходящего узла. У остальных столбцов наблюдаются выбросы и аномалии. Все ящики с усами для каждого столбца до и после обработки приведены в приложении 2.

1. Столбец с максимальным средним значением после нормировки признаков через стандартное отклонение?

Таковым столбцом является HostStarMassSlrMass - 9.195415701323054e-16.

1. Столбец с целевым признаком?

Такого столбца не имеется. У датасета нет столбца "Вид планеты". Все категории планет необходимо будет найти с помощью задачи кластеризации.

1. Сколько объектов попадает в тренировочную выборку при использовании train\_test\_split с параметрами test\_size = 0.3, random\_state = 42?

Так как задан параметр test\_size=0.3, то тренировочная выборка будет составлять 70% от всех данных (100%-30%=70%), т.е 2508 наблюдений.

1. Между какими признаками наблюдается линейная зависимость (корреляция)?

Построим корреляционную матрицу с помощью пакета Seaborn.

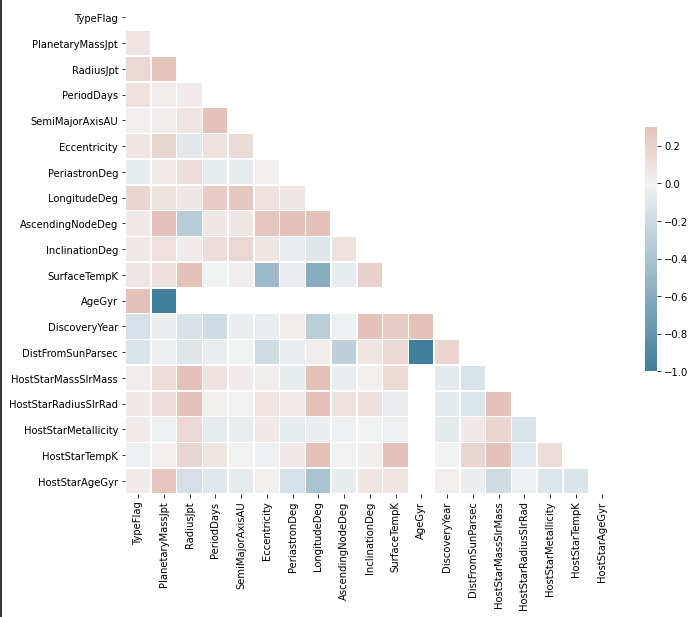


Рисунок 3. Корреляционная матрица выборки.

Можно заметить присутствие сильно негативно зависимых друг от друга данных (см. Рисунок 3). Так, например, существует сильная корреляция между AgeGyr и PlanetaryMassJpt, а также DistFromSunParsec, но так как PlanetaryMassJpt имеет всего лишь 2 значения, то можно проигнорировать данную корреляцию. Также заметим, что у некоторых столбцов пропущенные значения могут составлять больше половины всех наблюдений.

Исключим такие столбцы из матрицы корреляции посмотрим на новый результат.

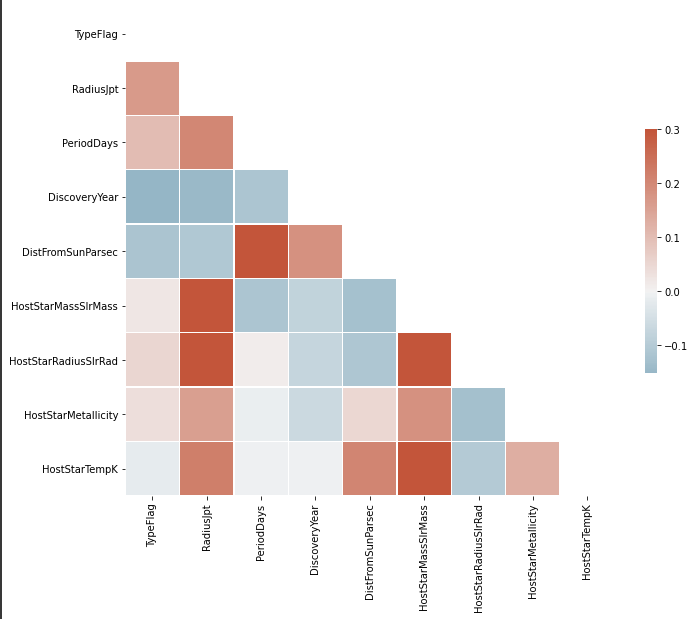


Рисунок 4. Корреляционная матрица выборки после удаления столбцов с более чем 50% пропущенных данных.

Из новой корреляционной матрицы (см. Рисунок 4) можно найти следующие зависимости:

* DistFromSunParsec и PeriodDays – действительно, чем дальше солнце, тем длиннее длится день.
* HostStarSlrMass и RadiusJpt – возможно, чем больше звезда, тем ей легче «притянуть» к себе большую планету.
* HostStarRadiusSlrRad и RadiusJpt – то же самое, что и пункт выше.
* HostStarRadiusSlrRad и HostStarSlrMass – чем больше звезда – тем больше ее масса.
* HostStarMassSlrMass и HostStarTempK - Чем массивнее звезда, тем больше ее температура.

1. Сколько признаков достаточно для объяснения 90% дисперсии после применения метода PCA?

Прежде чем приступить к построению PCA, данные следует обработать. Следует избавиться от тех столбцов, в которых пропущено более 50% данных, после чего разбить столбцы на несколько категориальных признаков или один бинарный, а столбцы с вещественными числами нормализовать.

Так, у столбца TypeFlag в выборке есть только "no known stellar binary companion", тогда можно применить one-hot encoding и получить один бинарный столбец. То же самое можно сделать для DiscoveryMethod – большинство объектов имеют тип «transit», и для ListsPlanetIsOn – большинство планет являются подтвержденными. Также, можно избавиться от года обнаружения и дополнения информации, так как такие столбцы не имеют значимой нагрузки. Также заметим, что у столбцов Declination и RightAscension есть 10 пропущенных объектов, 9 из которых планеты солнечной системы. Возможно, в последствии следует удалить эти 2 столбца, но на данный момент разделим их на категориальные признаки и заполним пропуски с помощью среднего значения. Остальные столбцы с действительными признаками отнормализуем.

Наконец, можно приступить к построению PCA.

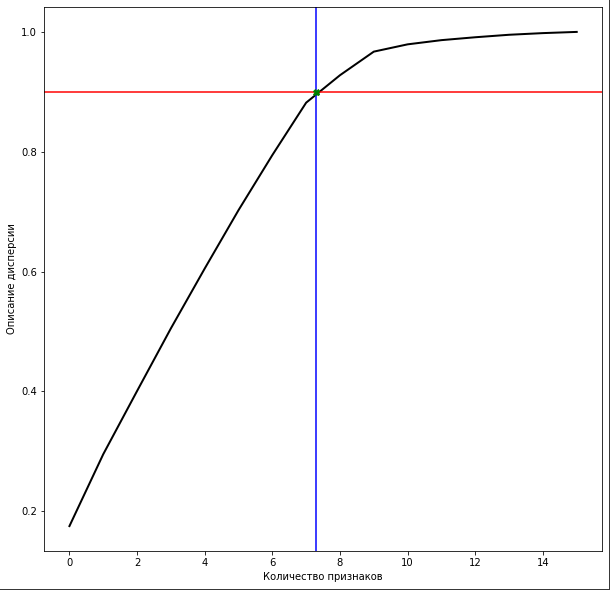


Рисунок 5.PCA выборки.

Как видно из рисунка 5, всего для описания более 90% дисперсии необходимо 8 признаков.

1. Какой признак вносит наибольший вклад в первую компоненту?

Dec1 вносит наибольший вклад: -0.632.

1. Построить двухмерное представление данных с помощью алгоритма t-SNE. На сколько кластеров визуально на ваш взгляд разделяется выборка?

Обучив t-SNE данные можно разделить примерно на 4 кластера, как это видно на рисунке 6.

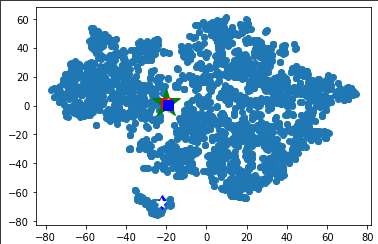


Рисунок 6. Представление данных с помощью t-SNE.

Подбирая параметры, можно прийти к более разделенной выборке, к которой уже модно применить алгоритм DBSCAN. На ней будет расположено уже 5 кластеров, однако планеты солнечной системы попадут в выбросы. Скорее всего, столбцы Declination и RightAscension следует удалить.

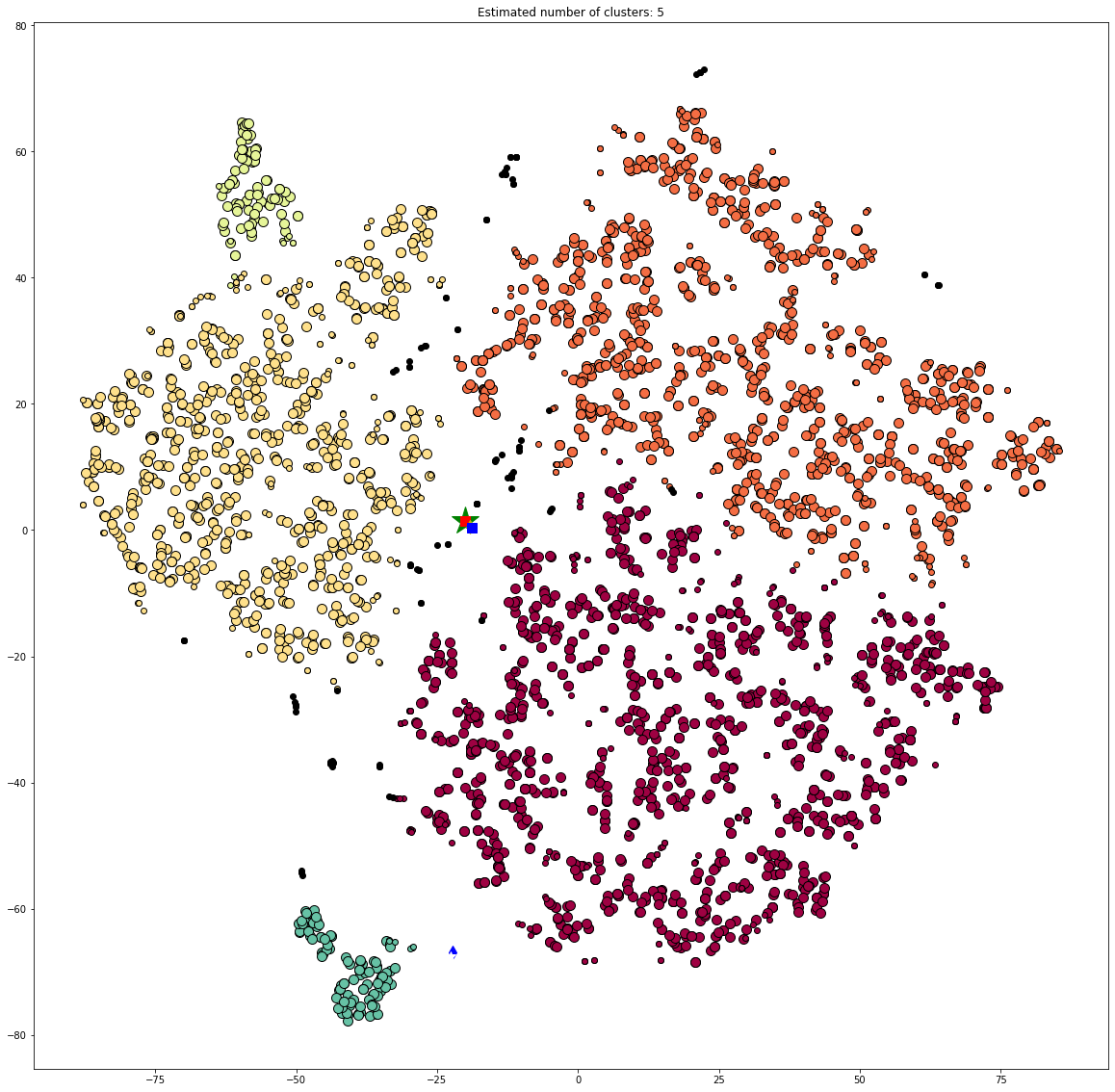


Рисунок 6. Представление данных с помощью t-SNE и BDSCAN.

Как можно увидеть,

Вывод

Не смотря на изначально обильное количество столбцов, из-за того, у многих столбцов отсутствует более 50% данных или они имеют слишком много аномалий из пригодных для дальнейшей работы остается лишь 16. Причем возможно, также следует избавиться от линейно зависимых друг от друга столбцов и таких столбцов, которые не содержат в себе данных только о солнечной системе. С помощью PCA удалось снизить размерность до 8 компонент, а DBSCAN на t-SNE обнаружил 5 различных типов планет, однако все планеты солнечной системы попали в «мусор», что требует более детального рассмотрения в 6 задаче.

Приложение 1

# -\*- coding: utf-8 -\*-

#!pip uninstall pandas-profiling

#!pip install pandas-profiling

# import pandas as pd

# import numpy as np

# import matplotlib.pyplot as plt

# import seaborn as sns

# import pandas\_profiling

"""Запустить следующий код в случае запуска с компьютера"""

#data = pd.read\_csv('oec.csv')

"""Запустить следующие две ячейки при использовании google colab"""

#from google.colab import drive

#drive.mount('/content/drive')

data = pd.read\_csv('/content/drive/MyDrive/oec.csv'

"""Посмотрим на предоставленные данные в датасете"""

from pandas\_profiling import ProfileReport

ProfileReport(data)

data

null = data[data.isna().any(axis=1)]

null

data.count().idxmin()

data.AgeGyr

data.AgeGyr.unique()

"""Больше всего пропусков в AgeGyr - из 3584 наблюдений лишь 2 имеют значение.

# 9. Есть ли на ваш взгляд выбросы, аномальные значения?

def moustachePlot(column):

f, axes = plt.subplots(2, 1)

sns.boxplot(column, palette="PRGn", ax=axes[0])

sns.distplot(column, ax=axes[1])

import warnings

warnings.filterwarnings("ignore")

moustachePlot(data.PlanetaryMassJpt)

mdata = data

mdata[mdata.PlanetaryMassJpt > 100]

mdata["PlanetaryMassJpt"][141] -= 90

mdata["PlanetaryMassJpt"][2905] -= 150

data = mdata

moustachePlot(data.PlanetaryMassJpt)

"""RadiusJpt"""

moustachePlot(data.RadiusJpt)

mdata = data

mdata[mdata.RadiusJpt > 2]

mdata["RadiusJpt"][1680] -= 3

data = mdata

moustachePlot(data.RadiusJpt)

moustachePlot(data.PeriodDays)

data['PeriodDays'].where(data['PeriodDays'] < 300000, 50000, inplace=True)

data.loc[data.PeriodDays > 300000, 'PeriodDays'] = data.loc[data.PeriodDays > 300000, 'PeriodDays'] - 25000

data.loc[data.PeriodDays > 50000, 'PeriodDays'] = data.loc[data.PeriodDays > 50000, 'PeriodDays'] - 50000

data.loc[data.PeriodDays > 140000, 'PeriodDays'] = 40000

data.loc[data.PeriodDays > 100000, 'PeriodDays'] = 30000

moustachePlot(data.PeriodDays)"""

moustachePlot(data.SemiMajorAxisAU)

data.loc[data.SemiMajorAxisAU > 50, 'SemiMajorAxisAU'] = data.loc[data.SemiMajorAxisAU > 50, 'SemiMajorAxisAU'] / 50

data.loc[data.SemiMajorAxisAU > 20, 'SemiMajorAxisAU'] = data.loc[data.SemiMajorAxisAU > 20, 'SemiMajorAxisAU'] / 20

data.loc[data.SemiMajorAxisAU > 5, 'SemiMajorAxisAU'] = data.loc[data.SemiMajorAxisAU > 5, 'SemiMajorAxisAU'] / 10

moustachePlot(data.SemiMajorAxisAU)

moustachePlot(data.Eccentricity)

data.loc[data.Eccentricity > 0.4, 'Eccentricity'] = data.loc[data.Eccentricity > 0.4, 'Eccentricity'] - 0.5

moustachePlot(data.Eccentricity)

moustachePlot(data.PeriastronDeg)

data.loc[data.PeriastronDeg < 200, 'PeriastronDeg'] = data.loc[data.PeriastronDeg < 200, 'PeriastronDeg'] + 100

data.loc[data.PeriastronDeg > 700, 'PeriastronDeg'] = data.loc[data.PeriastronDeg > 700, 'PeriastronDeg'] - 500

moustachePlot(data.PeriastronDeg)

moustachePlot(data.LongitudeDeg)

moustachePlot(data.AscendingNodeDeg)

moustachePlot(data.InclinationDeg)

data.loc[data.InclinationDeg < 70, 'InclinationDeg'] = data.loc[data.InclinationDeg < 70, 'InclinationDeg'] + 30

data.loc[data.InclinationDeg < 70, 'InclinationDeg'] = data.loc[data.InclinationDeg < 70, 'InclinationDeg'] + 30

data.loc[data.InclinationDeg < 70, 'InclinationDeg'] = data.loc[data.InclinationDeg < 70, 'InclinationDeg'] + 30

data.loc[data.InclinationDeg > 300, 'InclinationDeg'] = data.loc[data.InclinationDeg > 300, 'InclinationDeg'] - 160

moustachePlot(data.InclinationDeg)

moustachePlot(data.SurfaceTempK)

data.loc[data.SurfaceTempK > 5000, 'SurfaceTempK'] = data.loc[data.SurfaceTempK > 5000, 'SurfaceTempK'] - 4000

moustachePlot(data.SurfaceTempK)

moustachePlot(data.DistFromSunParsec)

moustachePlot(data.HostStarMassSlrMass)

moustachePlot(data.HostStarRadiusSlrRad)

moustachePlot(data.HostStarMetallicity)

moustachePlot(data.HostStarTempK)

data.loc[data.HostStarTempK > 15000, 'HostStarTempK'] = data.loc[data.HostStarTempK > 15000, 'HostStarTempK'] - 17000

moustachePlot(data.HostStarTempK)

moustachePlot(data.HostStarAgeGyr)

data.loc[data.HostStarAgeGyr > 50, 'HostStarAgeGyr'] = data.loc[data.HostStarAgeGyr > 50, 'HostStarAgeGyr'] - 75

moustachePlot(data.HostStarAgeGyr)

import copy

cols = [

data.PlanetaryMassJpt,

data.RadiusJpt,

data.PeriodDays,

data.SemiMajorAxisAU,

data.Eccentricity,

data.PeriastronDeg,

data.LongitudeDeg,

data.AscendingNodeDeg,

data.InclinationDeg,

data.SurfaceTempK,

data.DistFromSunParsec,

data.HostStarMassSlrMass,

data.HostStarRadiusSlrRad,

data.HostStarMetallicity,

data.HostStarTempK,

data.HostStarAgeGyr

]

cols = list(map(copy.deepcopy, cols))

cols[0].name

max = -999999

max\_name = None

for column in cols:

column = (column - column.mean()) / column.std()

cm = column.mean()

if cm > max:

max = cm

max\_name = column.name

print(max\_name, max)

data.shape[0]/100 \* 70

correlation = data.corr()

mask = np.triu(np.ones\_like(correlation, dtype=bool))

f, ax = plt.subplots(figsize=(11, 9))

cmap = sns.diverging\_palette(230, 20, as\_cmap=True)

sns.heatmap(correlation, mask=mask, cmap=cmap, vmax=.3, center=0,

square=True, linewidths=.5, cbar\_kws={"shrink": .5})

correlation = data.loc[:, data.count() >= 3584/2].corr()

mask = np.triu(np.ones\_like(correlation, dtype=bool))

f, ax = plt.subplots(figsize=(11, 9))

cmap = sns.diverging\_palette(230, 20, as\_cmap=True)

sns.heatmap(correlation, mask=mask, cmap=cmap, vmax=.3, center=0,

square=True, linewidths=.5, cbar\_kws={"shrink": .5})

data = data.drop(columns=data.loc[:, data.count() < 3584/2].columns)

data

data.set\_index('PlanetIdentifier', inplace=True)

data.TypeFlag.value\_counts()

data.TypeFlag = np.where(data.TypeFlag == 0, 1, 0)

data.TypeFlag.value\_counts()

data.DiscoveryMethod.value\_counts()

data.DiscoveryMethod = np.where(data.DiscoveryMethod == "transit", 1, 0)

data.DiscoveryMethod.value\_counts()

data = data.drop('DiscoveryYear', 1)

data = data.drop('LastUpdated', 1)

print(data.DistFromSunParsec.isna().sum())

data.loc[data.Declination.isna()]

data["Dec1"] = np.NaN

data["Dec2"] = np.NaN

data["Dec3"] = np.NaN

for i in range(data.shape[0]):

x = data['Declination'].iloc[i]

if x != x:

data.loc[data.index[i], 'Dec1'] = np.NaN

data.loc[data.index[i], 'Dec2'] = np.NaN

data.loc[data.index[i], 'Dec3'] = np.NaN

else:

x = x.split(' ')

data.loc[data.index[i], 'Dec1'] = float(x[0])

data.loc[data.index[i], 'Dec2'] = float(x[1])

data.loc[data.index[i], 'Dec3'] = float(x[2])

data = data.drop('Declination', 1)

data['Dec1'].fillna(value=np.mean(data['Dec1']),inplace=True)

data['Dec2'].fillna(value=np.mean(data['Dec2']),inplace=True)

data['Dec3'].fillna(value=np.mean(data['Dec3']),inplace=True)

data["Ra1"] = np.NaN

data["Ra2"] = np.NaN

data["Ra3"] = np.NaN

for i in range(data.shape[0]):

x = data['RightAscension'].iloc[i]

if x != x:

data.loc[data.index[i], 'Ra1'] = np.NaN

data.loc[data.index[i], 'Ra2'] = np.NaN

data.loc[data.index[i], 'Ra3'] = np.NaN

else:

x = x.split(' ')

data.loc[data.index[i], 'Ra1'] = float(x[0])

data.loc[data.index[i], 'Ra2'] = float(x[1])

data.loc[data.index[i], 'Ra3'] = float(x[2])

data['Ra1'].fillna(value=np.mean(data['Ra1']),inplace=True)

data['Ra2'].fillna(value=np.mean(data['Ra2']),inplace=True)

data['Ra3'].fillna(value=np.mean(data['Ra3']),inplace=True)

data = data.drop('RightAscension', 1)

data

data.ListsPlanetIsOn.value\_counts()

def conditions(x):

lst = [

"Confirmed planets",

"Confirmed planets, Planets in binary systems, S-type",

"Confirmed planets, Planets in binary systems, P-type",

"Confirmed planets, Planets in open clusters",

"Solar System", "Kepler Objects of Interest",

"Confirmed planets, Orphan planets",

"Planets in binary systems, S-type, Confirmed planets",

"Confirmed planets, Planets in binary systems, P-type, Planets in globular clusters"

]

if x in lst:

return 1

else:

return 0

func = np.vectorize(conditions)

data['ListsPlanetIsOn'] = func(data['ListsPlanetIsOn'])

data.ListsPlanetIsOn.value\_counts()

data.info()

data

data.PeriodDays = (data.PeriodDays - data.PeriodDays.mean()) / data.PeriodDays.std()

data.DistFromSunParsec = (data.DistFromSunParsec - data.DistFromSunParsec.mean()) / data.DistFromSunParsec.std()

data.HostStarRadiusSlrRad = (data.HostStarRadiusSlrRad - data.HostStarRadiusSlrRad.mean()) / data.HostStarRadiusSlrRad.std()

data.HostStarTempK = (data.HostStarTempK - data.HostStarTempK.mean()) / data.HostStarTempK.std()

data.Dec1 = (data.Dec1 - data.Dec1.mean()) / data.Dec1.std()

data.Dec2 = (data.Dec2 - data.Dec2.mean()) / data.Dec2.std()

data.Dec3 = (data.Dec3 - data.Dec3.mean()) / data.Dec3.std()

data.Ra1 = (data.Ra1 - data.Ra1.mean()) / data.Ra1.std()

data.Ra2 = (data.Ra2 - data.Ra2.mean()) / data.Ra2.std()

data.Ra3 = (data.Ra3 - data.Ra3.mean()) / data.Ra3.std()

data

data['RadiusJpt'].fillna(np.mean(data['RadiusJpt']), inplace=True)

data['PeriodDays'].fillna(np.mean(data['PeriodDays']), inplace=True)

data['DistFromSunParsec'].fillna(np.mean(data['DistFromSunParsec']), inplace=True)

data['HostStarMassSlrMass'].fillna(np.mean(data['HostStarMassSlrMass']), inplace=True)

data['HostStarRadiusSlrRad'].fillna(np.mean(data['HostStarRadiusSlrRad']), inplace=True)

data['HostStarMetallicity'].fillna(np.mean(data['HostStarMetallicity']), inplace=True)

data['Dec1'].fillna(np.mean(data['Dec1']), inplace=True)

data['Dec2'].fillna(np.mean(data['Dec2']), inplace=True)

data['Dec3'].fillna(np.mean(data['Dec3']), inplace=True)

data['Ra1'].fillna(np.mean(data['Ra1']), inplace=True)

data['Ra2'].fillna(np.mean(data['Ra2']), inplace=True)

data['Ra3'].fillna(np.mean(data['Ra3']), inplace=True)

data['HostStarTempK'].fillna(np.mean(data['HostStarTempK']), inplace=True)

data

from sklearn.decomposition import PCA

ProfileReport(data)

pca = PCA()

pca.fit(data)

X\_pca = pca.transform(data)

for i, component in enumerate(pca.components\_):

print("{} component: {}% of initial variance".format(i + 1,

round(100 \* pca.explained\_variance\_ratio\_[i], 2)))

print(" + ".join("%.3f x %s" % (value, name)

for value, name in zip(component, data.columns)))

plt.figure(figsize=(10,10))

plt.plot(np.cumsum(pca.explained\_variance\_ratio\_), color='k', lw=2)

plt.axhline(0.9, c='r')

plt.axvline(7.3, c='b')

plt.plot(7.3, 0.9, 'gX')

plt.xlabel('Количество признаков')

plt.ylabel('Описание дисперсии')

from sklearn.manifold import TSNE

tsne = TSNE(learning\_rate=100, random\_state=42)

transformed = tsne.fit\_transform(data)

def plotSolar(result, data):

earth = result[data.index.get\_loc("Earth")]

plt.plot(earth[0], earth[1], "g\*", markersize=30)

mercury = result[data.index.get\_loc("Mercury")]

plt.plot(mercury[0], mercury[1], "ws", markersize=10)

venus = result[data.index.get\_loc("Venus")]

plt.plot(venus[0], venus[1], "ys", markersize=10)

mars = result[data.index.get\_loc("Mars")]

plt.plot(mars[0], mars[1], "rs", markersize=10)

jupiter = result[data.index.get\_loc("Jupiter")]

plt.plot(jupiter[0], jupiter[1], "yd", markersize=10)

saturn = result[data.index.get\_loc("Saturn")]

plt.plot(saturn[0], saturn[1], "bs", markersize=10)

uranus = result[data.index.get\_loc("Uranus")]

plt.plot(uranus[0], uranus[1], "bd", markersize=10)

neptune = result[data.index.get\_loc("Neptune")]

plt.plot(neptune[0], neptune[1], "b\*", markersize=10)

pluto = result[data.index.get\_loc("Pluto")]

plt.plot(pluto[0], pluto[1], "w\*", markersize=10)

# Представляем результат в двумерных координатах

x\_axist = transformed[:, 0]

y\_axist = transformed[:, 1]

plt.scatter(x\_axist, y\_axist)

plotSolar(transformed, data)

plt.show()

perplexitys = [10, 20, 30]

rates = [20, 35, 50, 100]

models = []

for rate in rates:

for perp in perplexitys:

models.append(TSNE(learning\_rate=rate, perplexity=perp, random\_state=42))

results = []

for model in models:

transformed = model.fit\_transform(data)

results.append(transformed)

# Commented out IPython magic to ensure Python compatibility.

i = 1

# %matplotlib inline

for res in results:

plt.figure(figsize=(10,10))

x\_axist = res[:, 0]

y\_axist = res[:, 1]

plt.scatter(x\_axist, y\_axist)

plt.xlabel(f"Рисунок №{i}")

i+=1

plotSolar(res, data)

plt.show()

result = results[10]

plt.figure(figsize=(10,10))

x\_axist = result[:, 0]

y\_axist = result[:, 1]

plt.scatter(x\_axist, y\_axist)

i+=1

plt.show()

from sklearn.cluster import KMeans

inertia = []

for k in range(1, 20):

kmeans\_iter = KMeans(n\_clusters=k, random\_state=1).fit(data)

inertia.append(np.sqrt(kmeans\_iter.inertia\_))

plt.plot(range(1, 20), inertia, marker='s');

plt.xlabel("Число кластеров")

plt.ylabel("Cумма квадратов расстояний\nот точек до центроидов");

kmeans = KMeans(n\_clusters=8, random\_state=42)

kmeans.fit(data)

all\_predictions = kmeans.predict(data)

plt.figure(figsize=(10,10))

earth = result[data.index.get\_loc("Earth")]

plt.plot(earth[0], earth[1], "y\*", markersize=20)

plt.scatter(result[:, 0], result[:, 1], c=all\_predictions)

plotSolar(res, data)

plt.show()

"""DBSCAN"""

from sklearn.cluster import DBSCAN

db = DBSCAN(eps=5.6, min\_samples=17.5).fit(result)

core\_samples\_mask = np.zeros\_like(db.labels\_, dtype=bool)

core\_samples\_mask[db.core\_sample\_indices\_] = True

labels = db.labels\_

# Number of clusters in labels, ignoring noise if present.

n\_clusters\_ = len(set(labels)) - (1 if -1 in labels else 0)

n\_noise\_ = list(labels).count(-1)

print('Estimated number of clusters: %d' % n\_clusters\_)

print('Estimated number of noise points: %d' % n\_noise\_)

# Black removed and is used for noise instead.

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

unique\_labels = set(labels)

colors = [plt.cm.Spectral(each)

for each in np.linspace(0, 1, len(unique\_labels))]

X = result

for k, col in zip(unique\_labels, colors):

if k == -1:

# Black used for noise.

col = [0, 0, 0, 1]

class\_member\_mask = (labels == k)

xy = X[class\_member\_mask & core\_samples\_mask]

plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),

markeredgecolor='k', markersize=10)

xy = X[class\_member\_mask & ~core\_samples\_mask]

if result[data.index.get\_loc("Earth")] in xy:

print("EARTH HAS CLASS", k)

plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),

markeredgecolor='k', markersize=6)

plotSolar(res, data)

plt.title('Estimated number of clusters: %d' % n\_clusters\_)

plt.show()

Приложение 2

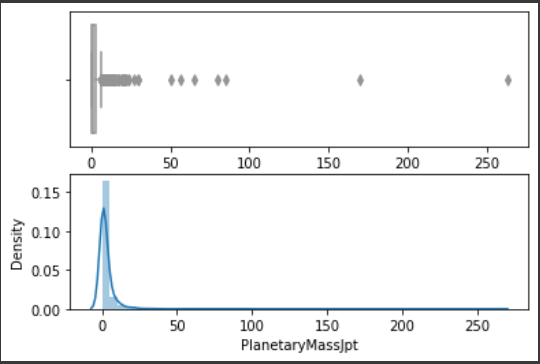


Рисунок 1. Ящик с усами для массы планеты.

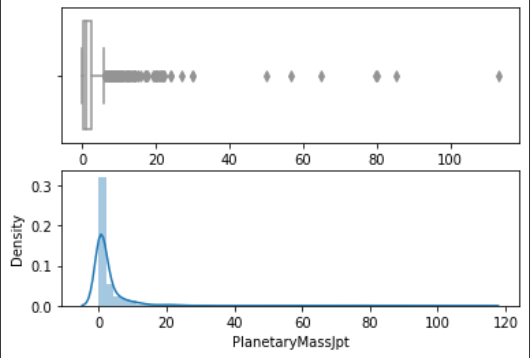


Рисунок 2. Ящик с усами для массы планеты после редактирования выбросов и аномалий.

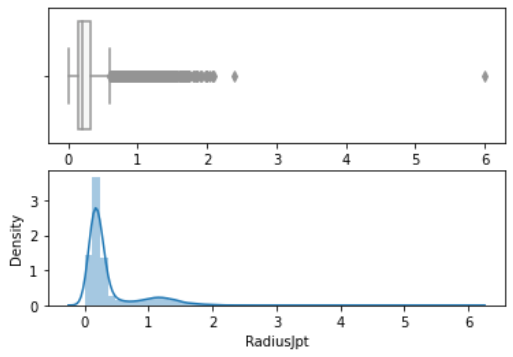


Рисунок 3. Ящик с усами для радиуса планеты.

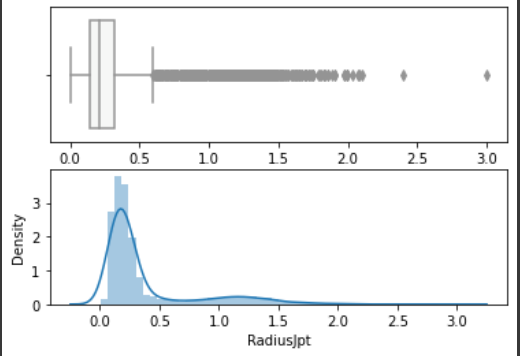


Рисунок 4. Ящик с усами для радиуса планеты после редактирования выбросов и аномалий.

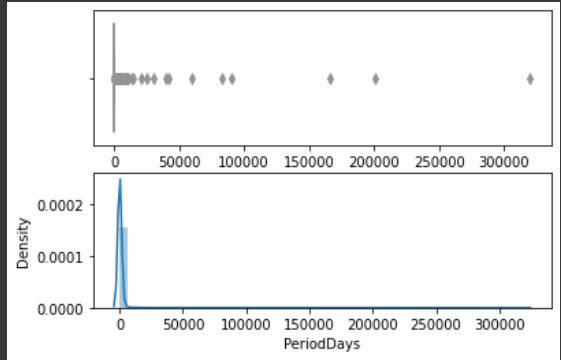


Рисунок 5. Ящик с усами для периода планеты.

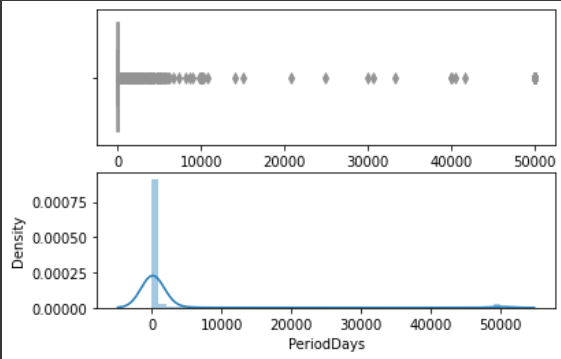


Рисунок 6. Ящик с усами для периода планеты после редактирования выбросов и аномалий.

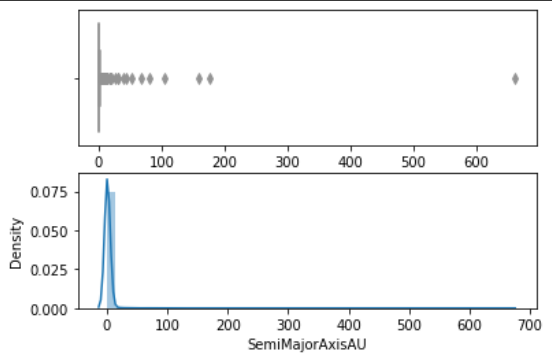


Рисунок 7. Ящик с усами для наклона планеты.

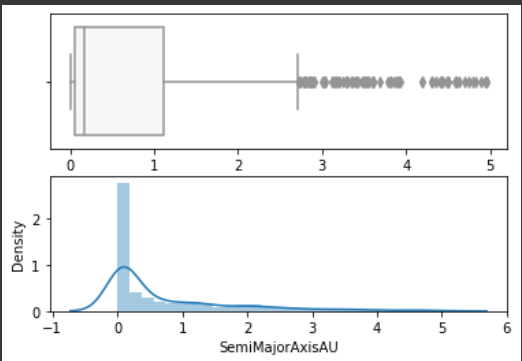


Рисунок 8. Ящик с усами для наклона планеты после редактирования выбросов и аномалий.

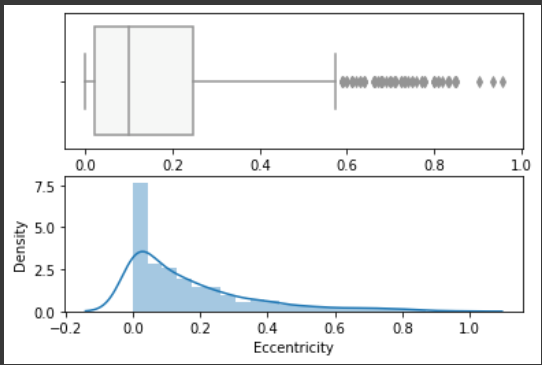


Рисунок 9. Ящик с усами для эксцентриситета планеты.

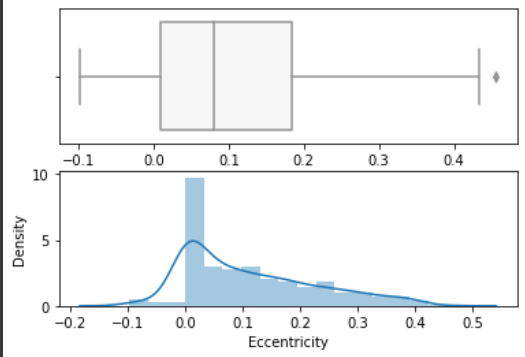


Рисунок 10. Ящик с усами для эксцентриситета планеты после редактирования выбросов и аномалий.

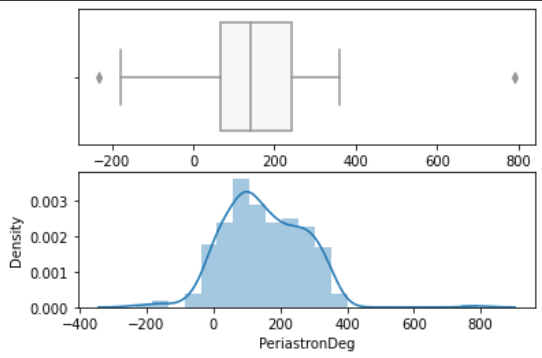


Рисунок 11. Ящик с усами для наклона периастра планеты.

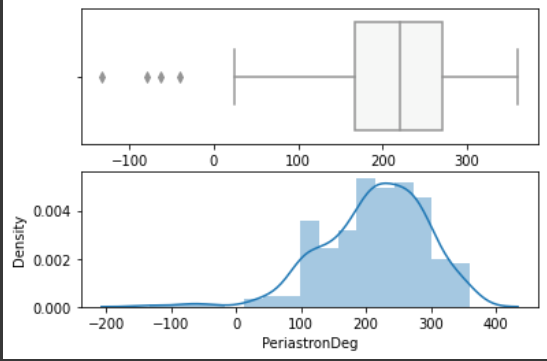


Рисунок 12. Ящик с усами для наклона периастра планеты после редактирования выбросов и аномалий.

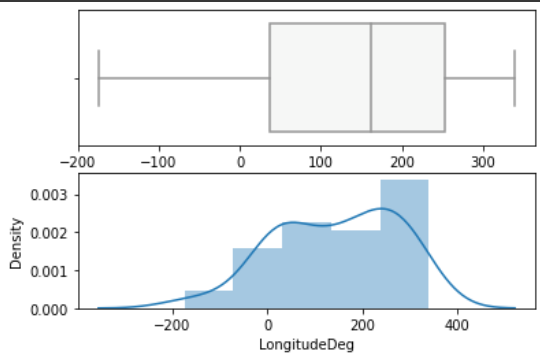


Рисунок 13. Ящик с усами для долготы планеты.

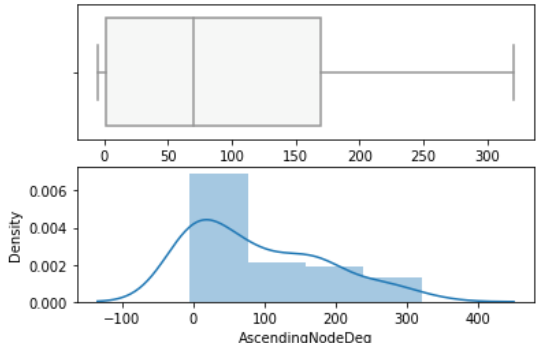


Рисунок 14. Ящик с усами для восходящего узла планеты.

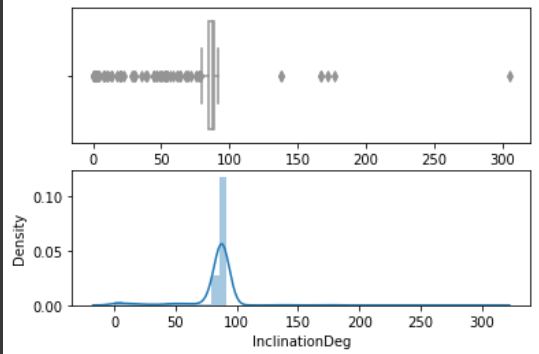


Рисунок 14. Ящик с усами для наклона планеты.

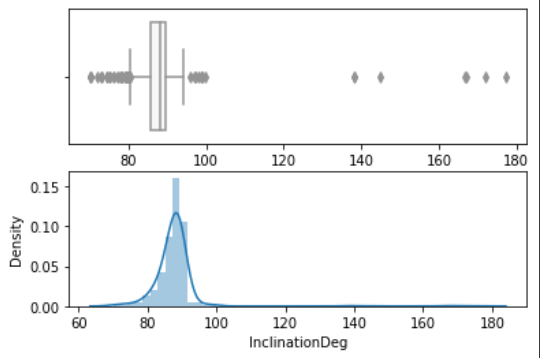


Рисунок 15. Ящик с усами для наклона планеты после редактирования выбросов и аномалий.

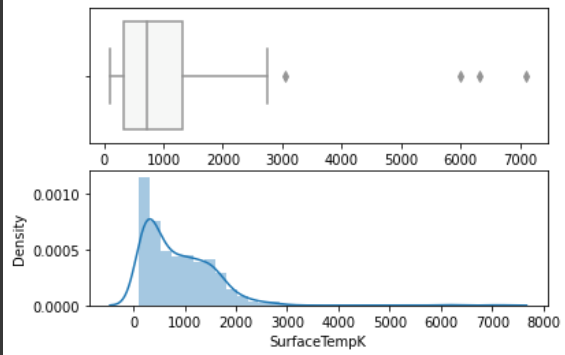


Рисунок 16. Ящик с усами для температуры планеты.

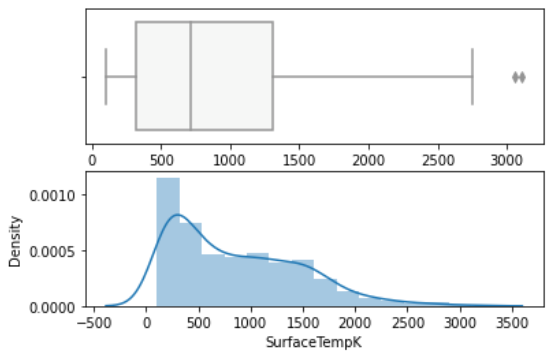


Рисунок 17. Ящик с усами для температуры планеты после редактирования выбросов и аномалий.

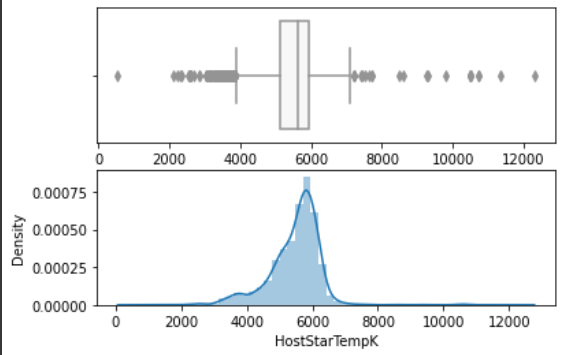


Рисунок 18. Ящик с усами для температуры звезды.

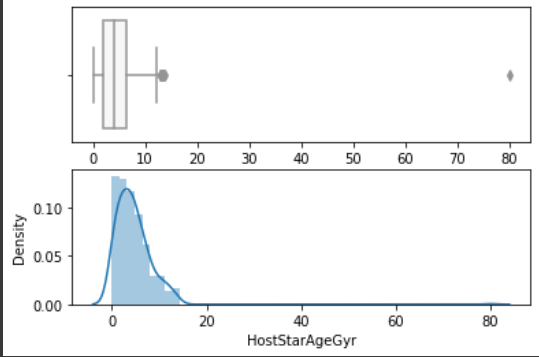


Рисунок 19. Ящик с усами для температуры звезды после редактирования выбросов и аномалий.

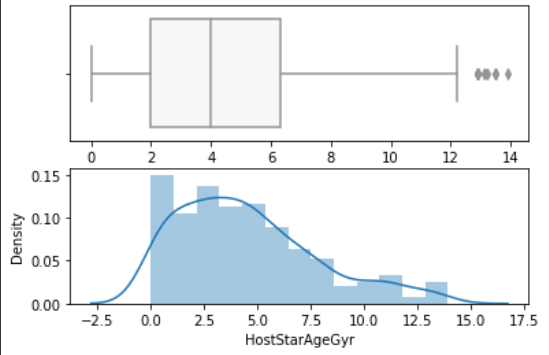


Рисунок 18. Ящик с усами для возраста звезды.