Оглавление

Задача 1

Необходимо загрузить данные из указанного набора и произвести следующие действия.

Набор данных: Swiss.

Объясняемая переменная: Education.

Регрессоры: Agriculture, Fertility.

1. Оценить среднее значение, дисперсию и СКО переменных, указанных во втором и в третьем столбце.

* Education.
  + Среднее – 10.98.
  + Дисперсия – 92.46.
  + СКО – 9.62.
* Argiculture.
  + Среднее – 50.66.
  + Дисперсия – 515.80.
  + СКО – 22.71.
* Fertility.
  + Среднее – 70.14.
  + Дисперсия – 156.04.
  + СКО – 12.49.

1. Построить зависимости вида y = a + bx, где y – объясняемая переменная, x – регрессор.

Таблица 1.1. Характеристики модели зависимости параметра Education от параметра Fertility в наборе данных Swiss.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Параметр \ Характеристики | Значение | Std. Error | t value | Pr(>|t|) | Уровень значимости |
| (Intercept) | 46.81788 | 6.11244 | 7.659 | 1.08e-09 | \*\*\* |
| Fertility | -0.51095 | 0.08582 | -5.954 | 3.66e-07 | \*\*\* |

Education = .

Судя по полученной модели (см. Таблица 1.1), чем выше рождаемость, тем хуже образование, что действительно похоже на правду – люди, которые получают образование, не имеют времени на создание семьи.

Таблица 1.2. Характеристики модели зависимости параметра Education от параметра Argiculture в наборе данных Swiss.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Параметр \ Характеристики | Значение | Std. Error | t value | Pr(>|t|) | Уровень значимости |
| (Intercept) | 24.69527 | 2.68890 | 9.184 | 6.98e-12 | \*\*\* |
| Argiculture | -0.27076 | 0.04852 | -5.580 | 1.30e-06 | \*\*\* |

Education =

По данной полученной модели (см. Таблица 1.2), чем выше количество занимающихся сельских хозяйством, тем хуже образование, что тоже является правдивыми данными – сельским хозяйством, в основном, занималось крестьянство.

1. Оценить, насколько «хороша» модель по коэффициенту детерминации .

Education =

. Модель плохо объясняет взаимосвязь, необходим ввод новых переменных.

Education =

. Модель плохо объясняет взаимосвязь, необходим ввод новых переменных.

1. Оценить, есть ли взаимосвязь между объясняемой переменной и объясняющей переменной.

Education =

При обоих коэффициентах значение p-статистики определённо является «хорошим» (см. Таблица 1.1), взаимосвязь между объясняемой переменной и регрессором действительно существует. Данная взаимосвязь объясняется следующим образом: чем больше у человека детей, тем меньше он может тратить времени на учебу.

Education =

При обоих коэффициентах значение p-статистики определённо является «хорошим» (см. Таблица 1.2), взаимосвязь между объясняемой переменной и регрессором действительно существует. Данная взаимосвязь объясняется следующим образом: если человек является крестьянином, то он вряд ли может позволить себе обучение.

Вывод по задаче 1

С помощью среды R [3] были построены две модели: Education от Agriculture и Education от Fertility. Обе полученные модели имеют действительно значимые элементы, а значит их можно адекватно описать, но имеют малый , поэтому необходимо вводить дополнительные переменные.

Существует взаимосвязь между Education и Fertility, а также между Education и Argiculture. Так, если у человека много детей, то он не имеет достаточно времени, чтобы тратить ее на учебу, а если человек является крестьянином, то у него недостаточно средств чтобы позволить себе обучение.

Задача 2

*Задача №2.1*

Необходимо загрузить данные из указанного набора и произвести следующие действия.

Набор данных: Swiss.

Объясняемая переменная: Infant.Mortality.

Регрессоры: Agriculture, Fertility, Education.

1. Проверьте, что в наборе данных нет линейной зависимости (построить зависимости между переменными, указанными в варианте, и проверить, что в каждой из них невысокий). В случае, если большой, один из таких столбцов можно исключить из рассмотрения.

Проверим линейную регрессию Catholic ~ Agriculture.

Таблица 2.1.1. Характеристики полученной модели Catholic ~ Agriculture.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Переменная | Estimate | Std. Error | t value | Pr(>|t|) | Уровень значимости |
| (Intercept) | 3.8313 | 13.8966 | 0.276 | 0.7840 |  |
| Agriculture | 0.7365 | 0.2508 | 22.71 | 0.0052 | \*\* |

= 16%. Значение p-статистики слабое - 2 звезды (см. Таблица 2.1.1). Принимая эти параметры во внимание, можно сказать, что линейная зависимость практически отсутствует. Можно использовать вместе в одной модели

Теперь проверим линейную регрессию Catholic ~ Education.

Таблица 2.1.2. Характеристики полученной модели Catholic ~ Education.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Переменная | Estimate | Std. Error | t value | Pr(>|t|) | Уровень значимости |
| (Intercept) | 48.4703 | 9.2805 | 5.223 | 4.37e-06 | \*\*\* |
| Education | -0.6673 | 0.6389 | -1.045 | 0.302 |  |

= 2%. Хоть уровень значимости у вспомогательной переменной и сильная (см. Таблица 2.1.2), но сама Education объясняет достаточно слабо. Зависимости совершенно не наблюдается, можно использовать в модели.

Наконец, проверим и линейную регрессию Agriculture ~ Education.

Таблица 2.1.3. Характеристики полученной модели Agriculture ~ Education.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Переменная | Estimate | Std. Error | t value | Pr(>|t|) | Уровень значимости |
| (Intercept) | 48.4703 | 9.2805 | 5.223 | 4.37e-06 | \*\*\* |
| Agriculture | -0.6673 | 0.6389 | -1.045 | 0.302 |  |

= 2%. Для данной зависимости наблюдаются те же значения (см. Таблица 2.1.3), что и у Catholic ~ Education. Зависимости совершенно не наблюдается, можно использовать в модели.

1. Построить линейную модель зависимой переменной (Infant.Mortality) от регрессоров (Agriculture, Fertility, Education) по методу наименьших квадратов. Оценить, насколько хороша модель, согласно: 1) , 2) p-значениям каждого коэффициента

Таблица 2.1.4. Характеристики полученной модели Infant.Mortality ~ Catholic+Agriculture+Education.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Параметр \ Характеристики | Значение | Std. Error | t value | Pr(>|t|) | Уровень значимости |
| (Intercept) | 22.36903 | 1.75389 | 12.754 | 3.29e-16 | \*\*\* |
| Catholic | 0.01904 | 0.01117 | 1.705 | 0.0954 | . |
| Agriculture | -0.04490 | 0.02636 | - 1.703 | 0.0957 | . |
| Education | -0.08520 | 0.05771 | -1.476 | 0.1472 |  |

10%. Значение низкое, а значит, нельзя сделать адекватные выводы по данной зависимости (см. Таблица 2.1.4). Также можно заметить, что p-значения у регрессоров очень высокие, т.е. Infant Mortality не зависит от них.

1. Ввести в модель логарифмы регрессоров. Сравнить модели и выбрать наилучшую.

Были построены и проверены следующие модели:

1. Infant Mortality ~ log(Catholic) + log(Agriculture) + log(Education)
2. log(drivers) ~ log(Catholic) + log(Agriculture) + log(Education) + Catholic + Agriculture + Education.

Значения для проверенных моделей:

1. 5% - p-значение у коэффициентов только увеличилось.
2. 14% - p-значение у коэффициентов только увеличилось.
3. Ввести в модель всевозможные произведения пар регрессоров, в том числе квадраты регрессоров. Найти одну или несколько наилучших моделей по доле объяснённого разброса в данных .

Для рассмотрения построим модель со всеми возможными парами регрессоров.

Таблица 2.1.5. Характеристики полученной модели Infant.Mortality ~ Catholic + Agriculture + Education + + + + Log(Catholic) + Log(Agriculture) + Log(Education) + (Catholic \* Agriculture) + (Catholic\*Education) + (Agriculture \* Education)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Параметр \ Характеристики | Значение | Std. Error | t value | Pr(>|t|) | Уровень значимости |
| (Intercept) | 18.0117962 | 1.427825 | 12.615 | 1.06e-15 |  |
| Catholic | -0.354176 | 0.263906 | -1.342 | 0.1885 |  |
| Agriculture | -0.154620 | 0.388734 | -0.398 | 0.6933 |  |
| Education | -0.250492 | 0.664041 | -0.377 | 0.7084 |  |
|  | 0.0006190 | 0.000384 | 1.612 | 0.11474 |  |
|  | 0.001007 | 0.002786 | 0.362 | 0.7199 |  |
|  | -0.0021146 | 0.0010389 | -2.035 | 0.04831 |  |
| Log(Catholic) | 1.2905177 | 0.7173768 | 1.799 | 0.07939 | . |
| Log(Agriculture) | 4.713491 | 5.382163 | 0.876 | 0.3873 |  |
| Log(Education) | 1.149989 | 2.646806 | 0.434 | 0.6667 |  |
| Catholic \* Agriculture | -0.0013712 | 0.0004525 | -3.031 | 0.00422 | . |
| Catholic \* Education | 0.002650 | 0.003251 | 0.815 | 0.4206 |  |
| Agriculture \* Education | -0.0003028 | 0.0015981 | -0.189 | 0.85066 |  |

полученной модели (см. Таблица 2.1.5) равен 33%, но уровень значимости всех параметров очень низкий. Проверим VIF данной модели, чтобы избавиться от наиболее незначимых параметров.

Таблица 2.1.6. VIF полученной модели со всеми возможными парами регрессоров.

|  |  |
| --- | --- |
| Параметр \ Характеристики | VIF |
| Catholic | 722.52177 |
| Agriculture | 464.90517 |
| Education | 243.16702 |
|  | 460.25190 |
|  | 223.76821 |
|  | 180.14396 |
| Log(Catholic) | 60.31557 |
| Log(Agriculture) | 101.51128 |
| Log(Education) | 26.30228 |
| Catholic \* Agriculture | 61.40025 |
| Catholic \* Education | 15.57974 |
| Agriculture \* Education | 19.76244 |

Наибольший VIF имеется у Catholic (см. Таблица 2.1.6), избавимся от него. После удаления модель получила =30%, а значит, удаление не сильно привело к потери качества модели. Продолжим избавляться от регрессоров с наибольшим VIF, пока не будет меняться слишком сильно.

Таблица 2.1.7. VIF полученной модели со всеми возможными парами регрессоров после удаления всех параметров, у которых был наибольший VIF.

|  |  |
| --- | --- |
| Параметр \ Характеристики | VIF |
|  | 19.984286 |
|  | 1.456378 |
| Log(Catholic) | 9.395438 |
| Catholic \* Agriculture | 29.270946 |
| Agriculture \* Education | 1.057898 |

При дальнейшем избавлении от регрессоров с наибольшим VIF (таблица 2.1.7) модели падал более чем на 5%. В итоге полученная модель имеет следующий вид.

Таблица 2.1.8. Характеристики полученной модели Infant.Mortality ~ + + Log(Catholic)+(Catholic \* Agriculture)+(Agriculture \* Education)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Параметр \ Характеристики | Значение | Std. Error | t value | Pr(>|t|) | Уровень значимости |
| (Intercept) | 18.0117962 | 1.427825 | 12.615 | 1.06e-15 | \*\*\* |
|  | 0.0006190 | 0.0003841 | 1.612 | 0.11474 |  |
|  | -0.0021146 | 0.0010389 | -2.035 | 0.04831 | \* |
| Log(Catholic) | 1.2905177 | 0.7173768 | 1.799 | 0.07939 | . |
| Catholic \* Agriculture | -0.0013712 | 0.0004525 | -3.031 | 0.00422 | \*\* |
| Agriculture \* Education | -0.0003028 | 0.0015981 | -0.189 | 0.85066 |  |

данной модели (см. Таблица 2.1.8) равен 26%.

Как итог, можно выделить две наилучшие модели:

А) Модель со всеми возможными парами регрессоров (см. Таблица 2.1.5) – Infant.Mortality ~ Catholic + Agriculture + Education + + + + Log(Catholic) + Log(Agriculture) + Log(Education) + (Catholic \* Agriculture) + (Catholic\*Education) + (Agriculture \* Education). Имеет неплохой, по сравнению с остальными моделями, , но все параметры совершенно не объясняют зависимость между ними и объясняемой переменной. Нельзя сделать какие-либо адекватные выводы.

Б) Infant Mortality ~ (см. Таблица 2.1.8). Хоть модели и ниже, чем у модели А, но появились регрессоры, по которым можно объяснить зависимость. Так, например, Catholic\*Agriculture действительно влияет на детскую смертность. Это можно объяснить тем, что более бедные регионы были сильнее подвержены католицизму, но если регион активно занимался фермерством, то у крестьянина было больше шансов заработать деньги и обратиться ко врачу в случае заболевания ребенка. Остальные регрессоры имеют незначительный вес, но в случае их удаления падал еще сильнее, поэтому их можно оставить.

Итог: После того, как мы убедились в том, что в наборе данных между объясняемой переменной и регрессорами нет линейной зависимости, были найдены две наиболее подходящие для работы модели. Первая такая модель является имеет все возможные пары регрессоров (см. Таблица 2.1.5) – у нее достаточно большой (по сравнению другими моделями) %, но ее регрессоры имеют очень малый уровень значимости. Тогда с помощью избавления от регрессоров с наибольшим VIF была получена вторая модель (см. Таблица 2.1.8), которая хоть и имеет меньший, чем у первой модели, но имеет более значимые переменные, которые уже возможно адекватно описать. В данной модели есть действительно значимая переменная Catholic\*Agriculture, и взаимосвязь данной переменной с Infant.Mortality можно описать следующим образом - более бедные регионы были сильнее подвержены католицизму, но если регион активно занимался фермерством, то у крестьянина было больше шансов заработать деньги и обратиться ко врачу в случае заболевания ребенка, отсюда и малая детская смертность в таком регионе.

*Задача №2.2*

1. Оцените доверительные интервалы для всех коэффициентов в модели, p=95%.

Всего имеется 40 степеней свободы в обучающей выборке. Было рассчитано 6 коэффициентов, тогда число степеней свободы в модели: 40–6=34. Отсюда можно подсчитать критерий Стьюдента, который будет равен .

Тогда доверительный интервал будет иметь следующий вид: , где:

* - значение коэффициента;
* t - значение t-критерия Стьюдента;
* σ- стандартная ошибка коэффициента в модели.

Подсчитаем доверительный интервал для каждого коэффициента и сделаем вывод о отвержении или невозможности отвергнуть статистическую гипотезу о том, что коэффициент равен 0.

Доверительный интервал для свободного коэффициента.

Середина интервала . СКО .

= =

0 не попадает в доверительный интервал. Коэффициент не может быть равен 0 на уровне значимости 5%.

Доверительный интервал для .

Доверительный интервал для свободного коэффициента.

Середина интервала . СКО .

= =

0 не попадает в доверительный интервал. Коэффициент не может быть равен 0 на уровне значимости 5%.

Доверительный интервал для .

Середина интервала . СКО .

= =

0 не попадает в доверительный интервал. Коэффициент не может быть равен 0 на уровне значимости 5%.

Доверительный интервал для .

Середина интервала . СКО .

= =

0 не попадает в доверительный интервал. Коэффициент не может быть равен 0 на уровне значимости 5%.

Доверительный интервал для .

Середина интервала . СКО .

= =

0 не попадает в доверительный интервал. Коэффициент не может быть равен 0 на уровне значимости 5%.

Доверительный интервал для .

Середина интервала . СКО .

= =

0 может попасть в доверительный интервал.

Таблица 2.2.1. Доверительные интервалы для всех регрессоров

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Регрессор | Оценка коэффициента перед регрессором | СКО | Доверительный интервал  (p = 95%, df=33, t=1.7) | Может ли = 0 |
| Свободный коэффициент | 16.6 | 2.1 | [13.03,20.17] | Нет |
|  | 0.0007 | 0.0004 | [0.00002,0.0014] | Нет |
|  | 0.0020607 | 0.0010427 | [−0.004,−0.0003] | Нет |
|  | 1.7047206 | 0.8494241 | [0.26,3.14] | Нет |
| Catholic×Argiculture | −0.0018518 | 0.0006937 | [−0.003,−0.0007] | Нет |
| Argiculture×Education | −0.0004987 | 0.0016156 | [−0.003,0.002] | Да |

Только в одном регрессоре может быть равен 0 – в Argiculture×Education (см. Таблица 2.2.1).

1. Оцените доверительный интервал для одного прогноза.

Доверительный интервал для прогноза . Построим датасет, взяв следующие данные из датасета Swiss:

1. Catholic – 10 объектов.
2. Agriculture – 20 объектов.
3. Education – 8 объектов.

Тогда прогноз модели ≈ 20.16, нижняя граница доверительного интервала ≈ 18.34, верхняя граница доверительного интервала ≈ 22.

Итог: помощью степеней свободы выборки, критерия Стьюдента и СКО для каждого регрессора нашли их доверительные интервалы (см. Таблица 2.2.1), причем у регрессора Argiculture×Education есть шанс, что 0 может попасть в доверительный интервал.

Вывод по задаче №2

*Задача №2.1*. Рассматриваемая модель была проверена на наличие линейной зависимости между регрессорами. Зависимостей обнаружено не было, и все переменные были использованы в модели. Однако при исследовании самой модели было выявлено, что объясняемая переменная (Infant.Mortality) практически не зависит от всех регрессоров.

В пункте №3 была попытка улучшить рассматриваемую модель, с помощью введения логарифмов регрессоров. Однако, не смотря на некоторый рост , точность все-еще является низкой.

В пункте №4 в модель были введены всевозможные произведения пар регрессоров, и была выявлена наилучшая модель по доле объяснённого разброса в данных . Найдена действительно значимая взаимосвязь между Catholic\*Agriculture и Infant.Mortality, которую можно описать следующим образом - более бедные регионы были сильнее подвержены католицизму, но если регион активно занимался фермерством, то у крестьянина было больше шансов заработать деньги и обратиться ко врачу в случае заболевания ребенка, отсюда и малая детская смертность в таком регионе.

*Задача №2.2*. Были найдены доверительные интервалы для всех коэффициентов в

рассматриваемой модели при p=95% и было выявлено, что из-за того, что значение коэффициента перед регрессором Argiculture×Education может быть равно 0, то данная объясняющая переменная практически не связана с объясняемой переменной Infant.Mortality.

В пункте №3 для оценивания доверительного интервала для одного прогноза, были

выбраны следующие значения:

1. Catholic – 10 объектов.
2. Agriculture – 20 объектов.
3. Education – 8 объектов.

Затем, с помощью функции predict() был вычислен прогноз ≈ 20.16 и доверительный интервал для рассматриваемой модели - нижняя граница доверительного интервала ≈ 18.34, верхняя граница доверительного интервала ≈ 22.

Задача 3

Необходимо проанализировать данные волны мониторинга экономического положения и здоровья населения РФ (данные обследования РМЭЗ НИУ ВШЭ).

Прежде чем начать выполнение задачи, были выбраны следующие столбцы:

1. Размер заработной платы – предсказываемая переменная.
2. Пол.
3. Семейное положение.
4. Образование.
5. Возраст.
6. Тип населенного пункта.
7. Часы работы в неделю.
8. Регион.
9. Удовлетворенность трудом.
10. Является ли иностранная фирма владельцем предприятия.
11. Имеет ли человек несколько работ.
12. Курит ли человек.

Далее были удалены строки, которые содержали в себе пустые значения, после чего началась подготовка выборки к работе. Так, для начала столбцы с действительными числами были отнормализованы – это столбец с зарплатой, возрастом, продолжительностью рабочей недели. Остальные же столбцы были приведены к бинарному или категориальному признаку:

1. Пол – бинарный признак.
2. Семейное положение – разделено на категориальный признак.
3. Образование – бинарный признак (наличие высшего образования).
4. Населенный пункт – бинарный признак (проживает ли человек в городе).
5. Регион – бинарный признак (проживает ли человек в одном из крупнейших регионов).
6. Удовлетворенность трудом – бинарный признак.
7. Является ли иностранная фирма владельцем предприятия – бинарный признак.
8. Имеет ли человек несколько работ – бинарный признак.
9. Курит ли человек – бинарный признак.
10. Постройте линейную регрессию зарплаты на все параметры, которые Вы выделили из данных мониторинга. Не забудьте оценить коэффициент вздутия дисперсии VIF.

Построим линейную регрессию salary ~ sex + wed1 + wed2 + wed3 + higher\_educ + age + status + dur + region + satisfy + foreign\_owner + second\_job + smokes

Таблица 3.1. Характеристики полученной линейной регрессии зарплаты.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Переменная | Estimate | Std. Error | t value | Pr(>|t|) | Уровень значимости |
| (Intercept) | -0.86107 | 0.07095 | -12.136 | 2e-16 | \*\*\* |
| sex | 0.47431 | 0.03248 | 14.603 | 2e-16 | \*\*\* |
| wed1 | 0.01475 | -0.07566 | -0.195 | 0.845482 |  |
| wed2 | 0.11767 | 0.06907 | 1.704 | 0.088515 | . |
| wed3 | 0.06779 | 0.07917 | 0.856 | 0.391924 |  |
| higher\_educ | 0.41977 | 0.03273 | 12.824 | 2e-16 | \*\*\* |
| age | -0.05684 | 0.01531 | -3.712 | 0.000208 | \*\*\* |
| status | 0.26169 | 0.03196 | 8.189 | 3.43e-16 | \*\*\* |
| dur | 0.13843 | 0.01421 | 9.739 | 2e-16 | \*\*\* |
| region | 0.57806 | 0.03318 | 17.422 | 2e-16 | \*\*\* |
| satisfy | 0.21698 | 0.02834 | 7.656 | 2.35e-14 | \*\*\* |
| foreign\_owner | 0.50696 | 0.07096 | 7.144 | 1.06e-12 | \*\*\* |
| second\_job | -0.07166 | 0.06627 | -1.081 | 0.279592 |  |
| smokes | 0.03496 | 0.03243 | 1.078 | 0.281011 |  |

слабый (см. Таблица 3.1), 24%, однако действительно имеются значимые регрессоры, которые можно объяснить. Проверим VIF для каждого регрессора.

Таблица 3.2. VIF полученной модели.

|  |  |
| --- | --- |
| Параметр \ Характеристики | VIF |
| sex | 1.404549 |
| wed1 | 5.666387 |
| wed2 | 6.052101 |
| wed3 | 2.876117 |
| higher\_educ | 1.094895 |
| age | 1.255599 |
| status | 1.068236 |
| dur | 1.081754 |
| region | 1.075931 |
| satisfy | 1.063886 |
| foreign\_owner | 1.014206 |
| second\_job | 1.008588 |
| smokes | 1.362337 |

Так как wed1 wed2 и wed3 имеют очень большой VIF (см. Таблица 3.2), а сами они не важны (низкий уровень значимости), то от них можно избавиться.

Таблица 3.3. Характеристики полученной модели после удаления wed1, wed2 и wed3.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Переменная | Estimate | Std. Error | t value | Pr(>|t|) | Уровень значимости |
| (Intercept) | -0.78582 | 0.03248 | -24.196 | 2e-16 | \*\*\* |
| sex | 0.48531 | 0.03160 | 15.357 | 2e-16 | \*\*\* |
| higher\_educ | 0.42763 | 0.03270 | 13.077 | 2e-16 | \*\*\* |
| age | -0.04380 | 0.01395 | -3.140 | 0.00017 | \*\* |
| status | 0.25938 | 0.03198 | 8.111 | 6.49e-16 | \*\*\* |
| dur | 0.13925 | 0.01423 | 9.758 | 2e-16 | \*\*\* |
| region | 0.57081 | 0.03318 | 17.203 | 2e-16 | \*\*\* |
| satisfy | 0.21437 | 0.02837 | 7.555 | 5.08e-14 | \*\*\* |
| foreign\_owner | 0.50685 | 0.07106 | 7.133 | 1.15e-12 | \*\*\* |
| second\_job | -0.07011 | 0.06637 | -1.056 | 0.2909 |  |
| smokes | 0.02842 | 0.03235 | 0.878 | 0.3797 |  |

новой модели практически не изменился (см. Таблица 3.3), но age стал менее значимым. Возможно, возраст респондента не сильно влияет на заработную плату.

1. Поэкспериментируйте с функциями вещественных параметров: используйте логарифм и степени (хотя бы от 0.1 до 2 с шагом 0.1).

Для нахождения наилучшей модели в среде R были использованы циклы. Были проверены следующие модели:

1. Степень age.
2. Степень dur.
3. Степень salary.
4. Степень age и dur.
5. Степень salary и dur.
6. Степень salary и age.
7. Все значения в степени.
8. Логарифм от age.
9. Логарифм от dur.
10. Логарифм от age и dur.
11. Логарифм от salary.

Наилучшей моделью оказалась та, которая имеет параметр и , 26%. Ввод еще одних логарифмов ни к чему не привел, а значит это и есть наилучшая модель.

Таблица 3.4. Характеристика новой модели со степенью 2 у age и dur.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Переменная | Estimate | Std. Error | t value | Pr(>|t|) | Уровень значимости |
| (Intercept) | -0.668916 | 0.034486 | -19.397 | 2e-16 | \*\*\* |
| sex | 0.506537 | 0.031378 | 16.143 | 2e-16 | \*\*\* |
| higher\_educ | 0.418646 | 0.032358 | 12.938 | 2e-16 | \*\*\* |
| age | -0.008808 | 0.014236 | -0.619 | 0.5362 |  |
|  | -0.119737 | 0.012373 | -9.677 | 2e-16 | \*\*\* |
| status | 0.264720 | 0.031648 | 8.364 | 2e-16 | \*\*\* |
| dur | 0.146109 | 0.017045 | 8.572 | 2e-16 | \*\*\* |
|  | -0.012027 | 0.006002 | -2.004 | 0.0452 | \* |
| region | 0.587792 | 0.032861 | 17.887 | 2e-16 | \*\*\* |
| satisfy | 0.235776 | 0.028144 | 8.378 | 2e-16 | \*\*\* |
| foreign\_owner | 0.485479 | 0.070311 | 6.905 | 5.76e-12 | \*\*\* |
| second\_job | -0.093516 | 0.065912 | -1.419 | 0.1560 |  |
| smokes | 0.007993 | 0.032061 | 0.249 | 0.8031 |  |

Проверим VIF полученной модели (см. Таблица 3.4).

Таблица 3.5. VIF новой модели.

|  |  |
| --- | --- |
| Параметр \ Характеристики | VIF |
| sex | 1.336076 |
| higher\_educ | 1.090584 |
| age | 1.106094 |
|  | 1.106598 |
| status | 1.067869 |
| dur | 1.585656 |
|  | 1.470079 |
| region | 1.075590 |
| satisfy | 1.069220 |
| foreign\_owner | 1.014818 |
| second\_job | 1.016841 |
| smokes | 1.357221 |

Хоть VIF у всех параметров мал (см. Таблица 3.5), но можно избавиться от малозначимых и посмотреть, что будет. Тогда избавимся от age и .

Таблица 3.6. Характеристика новой модели после удаления age и .

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Переменная | Estimate | Std. Error | t value | Pr(>|t|) | Уровень значимости |
| (Intercept) | -0.68 | 0.034374 | -20.176 | 2e-16 | \*\*\* |
| sex | 0.50837 | 0.03133 | 16.224 | 2e-16 | \*\*\* |
| higher\_educ | 0.41656 | 0.03232 | 12.888 | 2e-16 | \*\*\* |
|  | -0.12272 | 0.01199 | -10.235 | 2e-16 | \*\*\* |
| status | 0.26646 | 0.03164 | 8.420 | 2e-16 | \*\*\* |
| dur | 0.12740 | 0.01412 | 9.021 | 2e-16 | \*\*\* |
| region | 0.58700 | 0.03284 | 17.874 | 2e-16 | \*\*\* |
| satisfy | 0.23557 | 0.02814 | 8.371 | 2e-16 | \*\*\* |
| foreign\_owner | 0.48884 | 0.07027 | 6.957 | 4e-12 | \*\*\* |
| second\_job | -0.10490 | 0.06571 | -1.596 | 0.110 |  |
| smokes | 0.01238 | 0.03183 | 0.389 | 0.697 |  |

практическине изменился, а значит модель (см. Таблица 3.6) можно оставить в данном виде.

1. Сделайте вывод о том, какие индивиды получают наибольшую зарплату.

Судя по обученным коэффициентам модели, можно сделать следующие выводы:

1. Пол, к сожалению, влияет на зарплату опрошенных сотрудников. Мужчина получает больше, чем женщина.
2. Наличие высшего образования определенно влияет на зарплату.
3. Возраст отрицательно влияет на зарплату - чем старше или моложе человек относительно среднего (для предоставленных данных) возраста, тем больше шанс, что платить будут меньше.
4. Статус населенного пункта сказывается на зарплату.
5. Чем больше человек работает, тем больше (но не сильно больше) он получит денег
6. Регион определённо влияет на прибыль - в крупных регионах получают больше.
7. Если человек доволен работой, значит, скорее всего, он получает достаточную прибыль.
8. Иностранное предприятие принесет большую прибыль.
9. Если у человека несколько работ, значит, скорее всего, он получает недостаточную прибыль, из-за чего ему и нужно несколько работ.
10. Курение не сильно влияет на прибыль. Возможно, сигареты тогда стоили меньше, и не могли повлиять на бюджет.

Значит, следующий индивид может получить наибольшую зарплату:

* Имеет мужской пол.
* Имеет высшее образование.
* Среднего (из предоставленных данных) возраста.
* Из крупного региона.
* Работает на иностранное предприятие.

1. Оцените регрессии для подмножества индивидов, указанных в варианте.

Вариант 18, номер волны выборки РМЭЗ – 14.

Подмножество I - женщины, не замужем.

Таблица 3.7. Характеристика модели, построенная на подмножестве I.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Переменная | Estimate | Std. Error | t value | Pr(>|t|) | Уровень значимости |
| (Intercept) | -0.58028 | 0.04981 | -11.650 | 2e-16 | \*\*\* |
| higher\_educ | 0.39864 | 0.04805 | 8.296 | 3.39e-16 | \*\*\* |
|  | -0.08970 | 0.01719 | -5.217 | 2.20e-07 | \*\*\* |
| status | 0.15423 | 0.04897 | 3.149 | 0.00168 | \*\* |
| dur | 0.07373 | 0.02285 | 3.227 | 0.00129 | \*\* |
| region | 0.49552 | 0.04821 | 10.279 | 2e-16 | \*\*\* |
| satisfy | 0.10850 | 0.04257 | 2.549 | 0.01095 | \* |
| foreign\_owner | 0.80938 | 0.11880 | 6.813 | 1.64e-11 | \*\*\* |
| second\_job | -0.10688 | 0.09113 | -1.173 | 0.24115 |  |
| smokes | 0.06116 | 0.04769 | 1.283 | 0.19991 |  |

не сильно изменился (см. Таблица 3.7), но p-значение значительно ухудшилось - status, durability и satisfy объясняют хуже. Наибольшую зарплату получат женщины, которые имеют высшее образование, из крупного региона и работают на заграничное предприятие.

Подмножество II - женщины, живущие в городе, разведённые.

Таблица 3.8. Характеристика модели, построенная на подмножестве II.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Переменная | Estimate | Std. Error | t value | Pr(>|t|) | Уровень значимости |
| (Intercept) | -0.44978 | 0.07382 | -6.093 | 4.16e-09 | \*\*\* |
| higher\_educ | 0.24607 | 0.09556 | 2.575 | 0.0106 | \* |
|  | -0.10810 | 0.04410 | 2.451 | -0.0149 | \* |
| dur | 0.04055 | 0.05151 | 0.787 | 0.4319 |  |
| region | 0.53363 | 0.09858 | 5.413 | 1.45e-07 | \*\*\* |
| satisfy | 0.22352 | 0.08935 | 2.502 | 0.0130 | \* |
| foreign\_owner | 1.70699 | 0.26484 | 6.445 | 5.89e-10 | \*\*\* |
| second\_job | 0.11733 | 0.16337 | 0.718 | 0.4733 |  |
| smokes | 0.03601 | 0.09335 | 0.386 | 0.7000 |  |

стал лишь немного лучше, но p-значение осталось хорошим только у region и foreign owner (см. Таблица 3.8), из-за чего нельзя сделать адекватных выводов из полученной модели.

Вывод по задаче №3

Данные из 14 выборки были приведены к «хорошему» виду – строки с пустыми значениями были удалены, столбцы с действительными числами были отнормализованы, а остальные столбцы были приведены к бинарному или категориальному признаку. Далее, с помощью выбранных столбцов, которые содержат в себе информацию о поле, семейном положении, образовании, возрасте, типе населенного пункта, часов работы, регионе, удовлетворенности трудом, владельце предприятия, информации о имеет ли человек несколько работ и курит ли он была построена модель, которая предсказывает зарплату респондента (см. Таблица 3.1), однако данная модель имеет малый 24%, а некоторые регрессоры имеют большой VIF. Тогда с помощью вводов нелинейных переменных и с помощью избавления от переменных с большим VIF была получена новая модель (см. Таблица 3.8), которая имеет больший , а ее регрессоры можно адекватно описать. Из полученной модели следует, что в 2004 году наибольшую прибыль мог бы получить мужчина с высшим образованием, который имеет средний возраст и проживает в крупном регионе, а сам он работает на иностранное предприятие.

Затем с помощью тех же данных из 14 волны были выделены 2 подвыборки:

1. женщины, не замужем (см. Таблица 3.7);
2. женщины, живущие в городе, разведённые (см. Таблица 3.8):

Выборку I можно адекватно описать, так как большинство регрессоров являются значимыми. По полученной модели видно, что наибольшую зарплату получат женщины, которые имеют высшее образование, проживающие в крупном регионе и работают на заграничное предприятие. У выборки I имеется всего два значимых регрессора, а значит эту модель тяжело адекватно описать.

При этом стоит заметить, что во всех полученных моделях наиболее «сильными» критериями получения большей зарплаты, помимо пола, являются заграничный работодатель и регион, в котором проживает человек.

Задача 4

1. Обработайте набор данных набор данных, указанный во втором столбце таблицы 1, подготовив его к решению задачи классификации. Выделите целевой признак, указанный в последнем столбце таблицы, и удалите его из данных, на основе которых будет обучаться классификатор. Разделите набор данных на тестовую и обучающую выборку. Постройте классификатор типа, указанного в третьем столбце, для задачи классификации по параметру, указанному в последнем столбце. Оцените точность построенного классификатора с помощью метрик precision, recall и F1 на тестовой выборке.

Таблица 4.1. Набор данных.

|  |  |  |  |
| --- | --- | --- | --- |
| Вариант | Набор данных | Тип классификатора | Классификация по столбцу |
| 18 | Credit card customers | DecisionTreeClassifier | Customer Age (выше среднего значения – класс 0, ниже или совпадает – класс 1) |

После удаления столбца с целевым признаком (см. Таблица 4.1) необходимо привести данные к «хорошему» виду. Для начала, такие столбцы как “Gender”, “Attrition\_Flag”, “Education Level”, “Marital Status”, “Income Category” и “Card Category” были приведены к бинарному виду с помощью one hot encoding. Затем, столбцы с действительными числами “Customer Age”, “Months on book”, “Total Revolving Bal”, “Avg Open To Buy”, “Total Amt Chng Q4 Q1”, “Total Trans Amt”, “Total Trans Ct”, “Total Ct Chng Q4 Q1”, “Credit Limit” были также ортнормированы.

Далее, столбец Customer Age был промаркирован в соответствии с требованием задачи – если значение больше 0, т.е. больше среднего значения в неотнормированной выборке, то данная строка маркируется как 0, иначе маркируется как 1.

Наконец, выборка была поделена на тренировочную (77%) и тестовую (33%), а на ее основе с помощью пакета sklearn [2] был обучен первый классификатор без подбора параметров. Данный классификатор имеет глубину 4 и 16 листьев.

Рисунок 4.1. Обученное дерево решений высотой 4.

Судя по полученному классификатору (см. Рисунок 4.1), если человек зарегистрирован в банковской системе был недавно, или же у него зарегистрировано много доходов на иждивении и у него меньше транзакций, то данный человек имеет возраст меньше среднего. Проверим точность построенного классификатора.

Таблица 4.2. Точность полученного классификатора.

|  |  |
| --- | --- |
| Метрика | Оценка |
| f1 | 78% |
| precision | 76% |
| recall | 82% |

Точность полученного классификатора (см. Таблица 4.2) приемлемая.

Попробуем подобрать гиперпараметры с помощью Grid Search чтобы получить более точную модель. Пускай минимальная глубина варьируется от 2 до 30, а выборку можно будет разбить от 2 до 10 подвыборок.

После обучения было получено дерево высотой 3.

Рисунок 4.2. Обученное дерево с помощью Grid Search высотой 4.

Судя по полученному классификатору (см. Рисунок 4.2), если человек был зарегистрирован недавно, или он имеет много доходов на иждивении, то его возраст меньше среднего. Также проверим точность классификатора.

Таблица 4.3. Точность полученного классификатора с использованием Grid Search.

|  |  |
| --- | --- |
| Метрика | Оценка |
| f1 | 79% |
| precision | 74% |
| recall | 85% |

Полученный классификатор высотой 3 (см. Таблица 4.3) имеет схожую точность с классификатором высотой 4 (см. таблица 4.2).

1. Постройте классификатор типа Случайный Лес (Random Forest) для решения той же задачи классификации. Оцените его качество с помощью метрик precision, recall и F1 на тестовой выборке. Какой из классификаторов оказывается лучше?

Обучим случайный лес со следующими параметрами:

* Возможное число деревьев – от 5 до 400.
* Максимальная глубина – от 2 до 20
* Критерий обучения – критерий Джини.

После обучения была получена модель из 100 деревьев. Проверим точность данной модели.

Таблица 4.3. Точность полученного классификатора с использованием случайного леса.

|  |  |
| --- | --- |
| Метрика | Оценка |
| f1 | 80% |
| precision | 75% |
| recall | 85% |

Точность данной модели (см. таблица 4.3) из 100 деревьев практически не превосходит модель из одного дерева с 3 вершинами.

Вывод по задаче 4

После обработки данных построили три различных классификатора – обычный классификатор высотой 4, классификатор с подбором гиперпараметров в Grid Search и классификатор с использованием случайного леса. Все три классификатора имеют примерно одинаковую точность, поэтому разумнее всего использовать наименьший по высоте и количеству деревьев классификатор, а именно один классификатор высотой 3.

Судя по данному классификатору, с вероятностью в ~79% можно определить, имеет ли человек возраст меньше среднего лишь по двум переменным – как долго существует его банковский аккаунт и по количеству доходов на иждивении.

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 5.1. 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 [2].

Так, ящик с усами для массы планеты представлен на Рисунке 5.1:

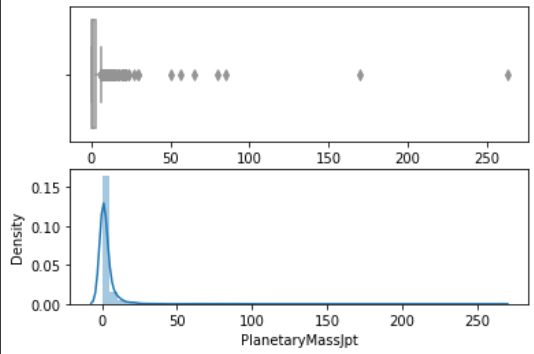


Рисунок 5.. Ящик с усами для массы планет.

Планеты, выходящие за ящик с усами, действительно являются аномалиями, но из-за того, что некоторые значения действительно большие, их стоит сократить. Результат после обработки предоставлен на Рисунке 5.2.

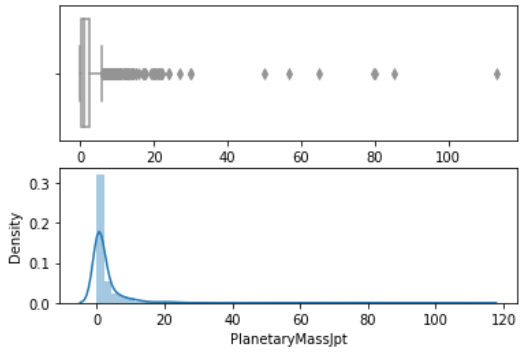


Рисунок 5.2. Ящик с усами для массы планет после изменения некоторых аномальных значений.

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 5.2.

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 [1, c. 378].

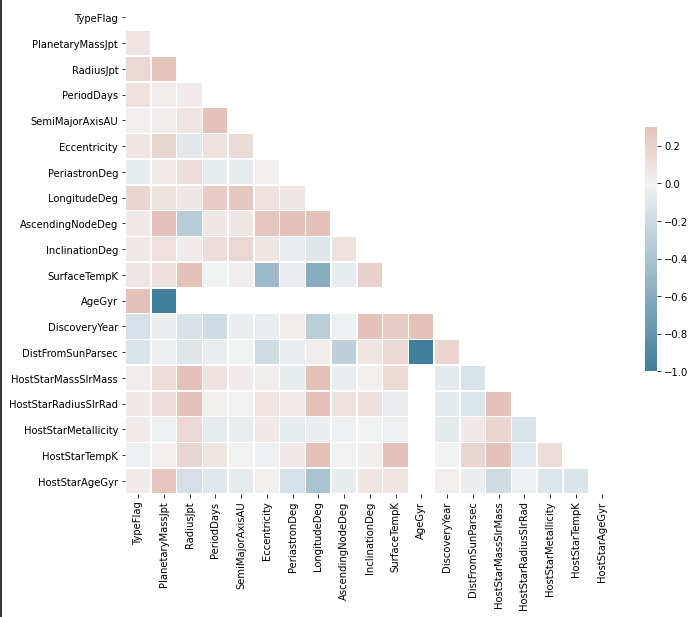


Рисунок 5.3. Корреляционная матрица выборки.

Можно заметить присутствие сильно негативно зависимых друг от друга данных (см. Рисунок 5.3). 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.

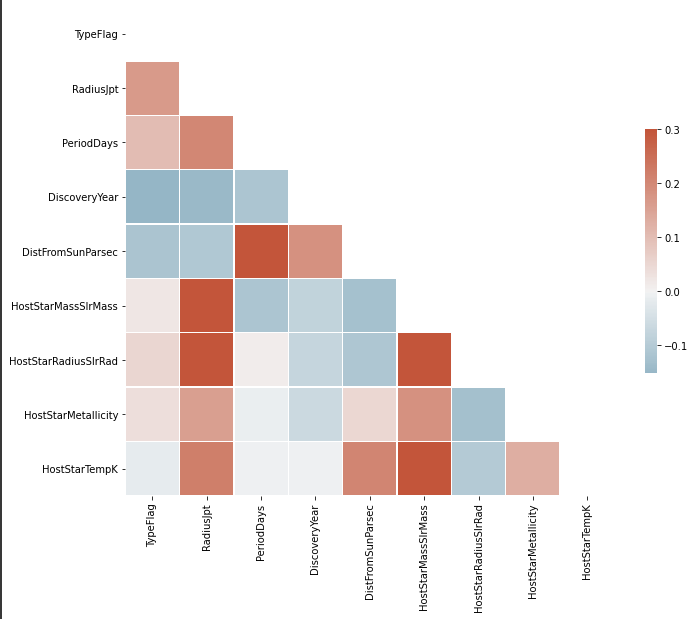


Рисунок 5.4. Корреляционная матрица выборки после удаления столбцов с более чем 50% пропущенных данных.

Из новой корреляционной матрицы (см. Рисунок 5.4) 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.

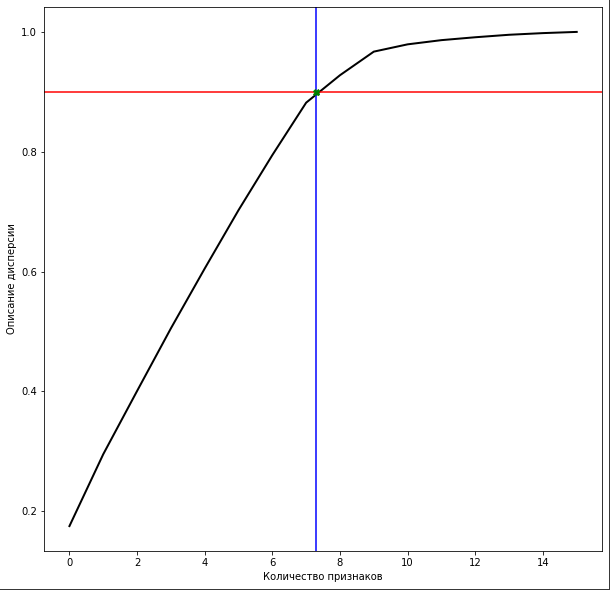


Рисунок 5.5.PCA выборки.

Как видно из Рисунка 5.5, 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 5.6.

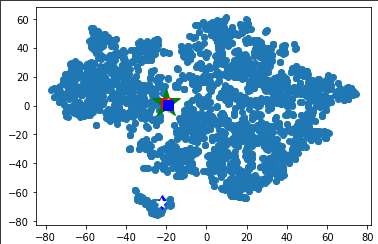


Рисунок 5.6. Представление данных с помощью t-SNE.

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.

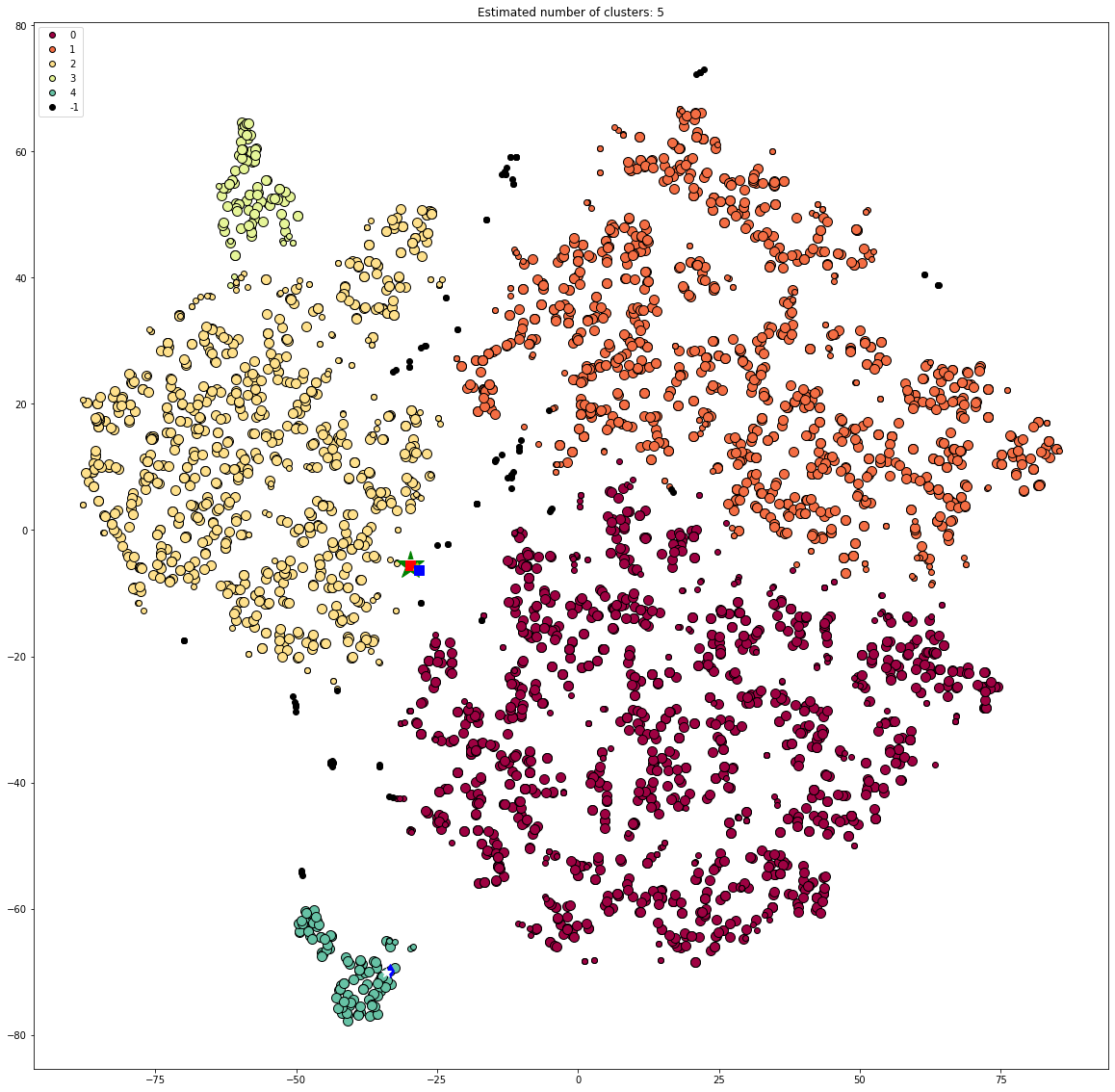


Рисунок 5.7. Представление данных с помощью t-SNE и BDSCAN.

Рассмотрим каждый кластер (см. Рисунок 5.7) отдельно.

* Кластер 0.
  + RadiusJpt – от 0.002 до 2.1. Среднее 0.28.
  + PeriodDays – от 0.17 до 3538. Среднее 50.
  + DistFromSunParsec – от 4.5 до 3500. Среднее 738.
  + HostStarMassSlrMass – от 0.06 до 2.3. Среднее 1.
  + HostStarRadiusSlrRad – от 0.2 до 6.3. Среднее 1.
  + HostStarMetallicity – от -0.57 до 0.48. Среднее 0.01.
  + HostStarTempK – от 3208 до 29300. Среднее 5663.
* Кластер 1.
  + RadiusJpt – от 0.05 до 2. Среднее 0.26.
  + PeriodDays – от 0.35 до 9886. Среднее 72.
  + DistFromSunParsec – от 14 до 3200. Среднее 705.
  + HostStarMassSlrMass – от 0.13 до 3. Среднее 1.
  + HostStarRadiusSlrRad – от 0.2 до 5. Среднее 1.
  + HostStarMetallicity – от -.1 до 0.5. Среднее 0.01.
  + HostStarTempK – от 3157до 7717. Среднее 5522.
* Кластер 2 – наиболее ближайшие к солнцу планеты.
  + RadiusJpt – от 0.005 до 2. Среднее 1.
  + PeriodDays – от 0.09 до 15000. Среднее 629.
  + DistFromSunParsec – от 1.295 до 3040. Среднее 170.
  + HostStarMassSlrMass – от 0.02 до 4. Среднее 1.
  + HostStarRadiusSlrRad – от ~0 до 8. Среднее 1.3.
  + HostStarMetallicity – от -2 до 0.5. Среднее 0.03.
  + HostStarTempK – от 540 до 9790. Среднее 5392.
* Кластер 3 – планеты с неизвестным радиусом.
  + PeriodDays – от 6.495 до 3186. Среднее 629.
  + DistFromSunParsec – от 10.34 до 759. Среднее 170.
  + HostStarMassSlrMass – от 0.74 до 4.5. Среднее 2.
  + HostStarRadiusSlrRad – от 3.8 до 51. Среднее 15.
  + HostStarMetallicity – от - 0.74 до 0.4. Среднее -0.08.
  + HostStarTempK – от 3950 до 5242. Среднее 4692.
* Кластер 4 – наиболее удаленные планеты от ближайшей звезды.
  + RadiusJpt – от 0.001 до 6. Среднее 1.5.
  + PeriodDays – от 20816 до 320000. Среднее 101916.
  + DistFromSunParsec – от 1.3 до 8200. Среднее 1995.
  + HostStarMassSlrMass – 0.01 до 2.8. Среднее 0.8.
  + HostStarRadiusSlrRad – от 0.2 до 4.22. Среднее 15.
  + HostStarMetallicity – от - 0.6 до 0.4. Среднее -0.05.
  + HostStarTempK – от 2100 до 11361. Среднее 5551.

Вывод по задаче 5

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. Кластеры разделяются на наиболее удаленные от солнца, на наиболее близкие к солнцу и, так как кластер 0 и кластер 1 похожи друг на друга – «средние» по удаленности от солнца планеты. Также существует отдельный кластер, в котором у всех планет отсутствуют данные о их радиусе.

Задача 6

В связи объемом предстоящей работы из-за необходимости кластеризации и классификации данных, данная задача была поделена – в данной работе будет рассмотрена кластеризация данных из задачи 5, а в работе Солониченко Златы будет проведена классификация полученных планет.

В предыдущей задаче было предположено, что столбцы RightAscension и Declination мешают кластеризации и для этого их необходимо удалить. Создадим такой датасет без данных столбцов. После чего снова проведем снижение размерности с помощью PCA.

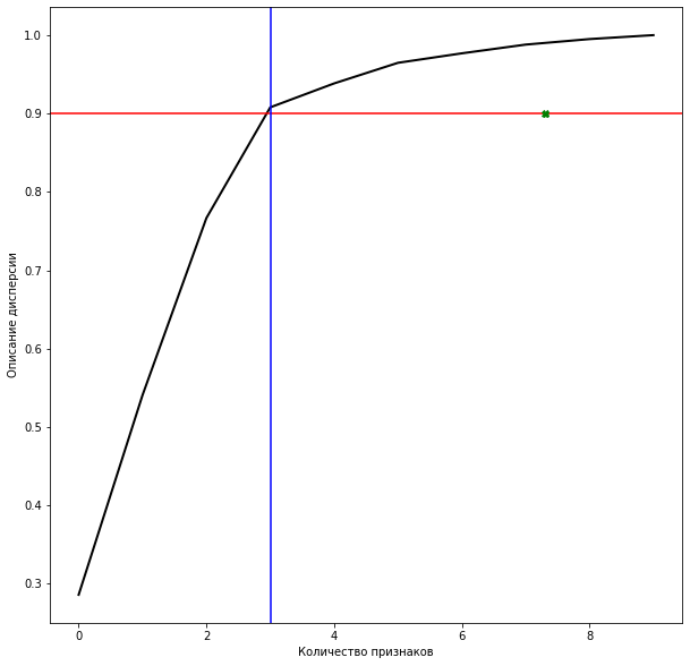


Рисунок 6.1. PCA выборки.

Как можно заметить, без данных столбцов выборку удалось описать всего лишь в трех компонентах (см. Рисунок 6.1). Посмотрим, как расположены данные в трехмерном пространстве.

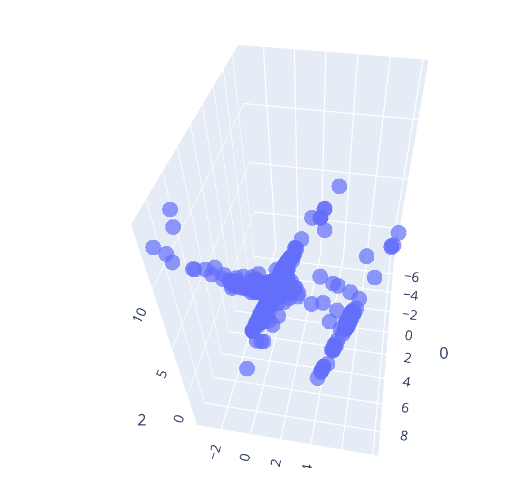


Рисунок 6.2. Расположение данных в трехмерном пространстве после понижения размерности.

Видно, что данные можно разделить на несколько кластеров (см. Рисунок 6.2). Для получения кластеров используем метод ближайших соседей.

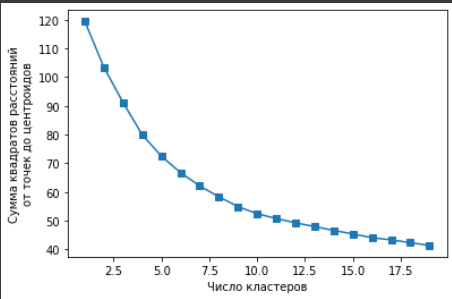


Рисунок 6.3. Нахождение оптимального количества числа кластеров.

Как видно из Рисунка 6.3, оптимальным количеством кластеров будет являться 10 штук. Тогда применим метод ближайших соседей из 10 кластеров.

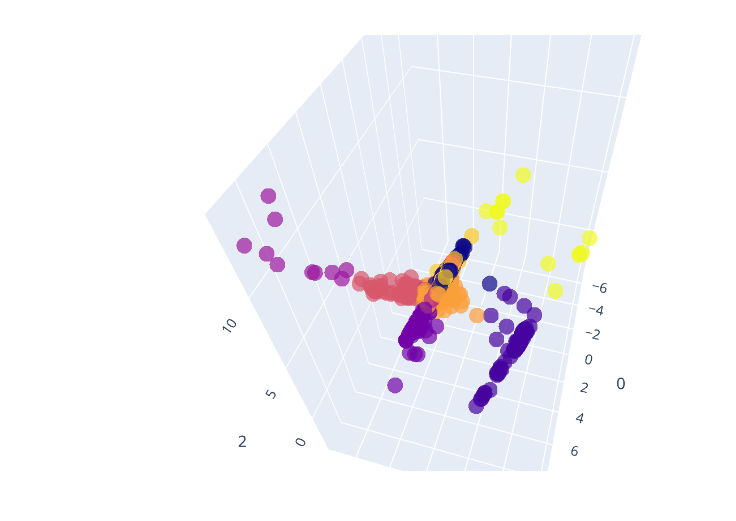


Рисунок 6.4. Данные после кластеризации с помощью метода ближайших соседей.

Наконец, проверим в какие кластеры (см. Рисунок 6.4) попали планеты солнечной системы.

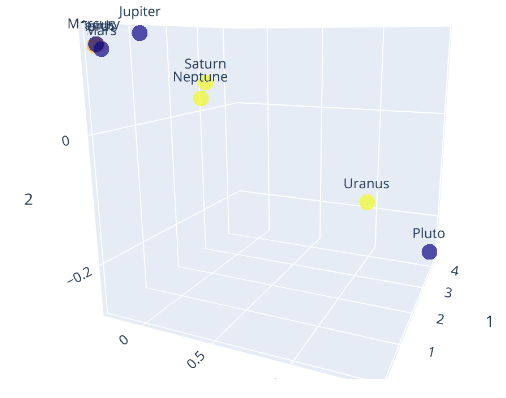


Рисунок 6.5. Планеты солнечной системы после кластеризации.

Из Рисунка 6.5 видно, что планеты солнечной системы попали в следующие кластеры:

* Кластер 2.
  + Венера.
  + Марс.
  + Плутон.
* Кластер 7
  + Земля.
  + Меркурий.
* Кластер 8
  + Сатурн.
  + Уран.
  + Нептун.

Рассмотрим каждый кластер подробнее.

* Кластер №0 - состоит из 307 наблюдений, больше остальных планет в среднем на 100%, при этом находятся к ближайшей звезде ближе, чем остальные.
* Кластер №1 – состоит из 101 наблюдения, на 50% больше среднего размера планеты в выборке, находятся на среднем расстоянии от планеты.
* Кластер №2 – состоит из 252 наблюдений, в основном из планет на 20% больше, чем остальные планеты в среднем.
* Кластер №3 – состоит из 10 наблюдений, у которых отсутствует информация про их размер. Такие планеты также находятся ближе к ближайшей звезде, чем в среднем.
* Кластер №4 – состоит из 705 наблюдений, большинство которых было сделано телескопом «Кеплер». На 30% больше среднего значения по размеру, находятся от ближайшей звезды дальше, чем остальные.
* Кластер №5 – состоит из 63 наблюдений, не имеют информации о размере планеты, находятся на 60% ближе к солнцу, чем остальные планеты в среднем.
* Кластер №6 – состоит из 443 наблюдений, большинство открыты телескопом «Кеплер», на 32% больше, чем остальные планеты в среднем. Находятся в среднем в 2 раза дальше от ближайшей звезды, чем остальные планеты.
* Кластер №7 – 564 наблюдения, имеют средний размер и находятся достаточно близко к ближайшей звезде.
* Кластер №8 – 1129 наблюдений, на 20% больше, чем остальные планеты в среднем. Находятся немного дальше от звезды, чем остальные.
* Кластер №9 – 14 наблюдений, больше остальных планет в среднем на 70%, также удалены от ближайшей звезды на 20%.

Также рассмотрим кластер, в который попала Земля. Так, при более детальном рассмотрении можно заметить, что большинство экзоплнет из данного кластера находятся в системе, похожей на солнечную, при этом в данный кластер также попали и кандидаты на планеты пригодные для жизни, например - Tau Ceti f и HD 40307 g. [5]

Вывод по задаче 6

С помощью понижения размерности до трех компонент и последующей кластеризации с методом ближайших соседей неразмеченный датасет удалось разделить на 10 кластеров, причем в отличие от полученной кластеризации в задаче 5 планеты не находятся в одной точке, а расположены в разных кластерах. Так, кластеры смогли разделиться по размеру и отдаленности от звезды, а также по типу звезды. При этом появился кластер, в котором можно найти потенциальных кандидатов экзопланет, пригодных к жизни – в этот кластер входит и сама Земля.

Заключение

В ходе данной работы были завершены поставленные задачи по исследованию зависимостей построенных моделей, построению классификаторов вида решающее дерево, случайный лес и подбор их гиперпараметров с помощью GridSearch. Также была проведена обработка данных и их последующая кластеризация.

Так, в задаче 1 были построены две зависимости с объясняемой переменной

Education от регрессоров Fertility и Agriculture. Данные модели имеют высокий уровень значимости, а значит между регрессорами и объясняемой переменной есть связь, но имеют низкий уровень модели, а поэтому требуют ввода дополнительных переменных. Из взаимосвязей можно сделать следующий вывод:

* 1. Если у человека много детей, то он не имеет достаточно времени, чтобы тратить ее на учебу.
  2. Если человек является крестьянином, то у него недостаточно средств чтобы позволить себе обучение.

В задаче 2 была рассмотрена модель зависимости Infant.Mortality~ Agriculture, Fertility, Education. Было обнаружено, что уровень значимости регрессоров очень низкий, и поэтому между ними и объясняемой переменной Infant.Mortality не существует связи. После ввода всевозможных произведений пар регрессоров была найдена действительно значимая взаимосвязь между Catholic\*Agriculture и Infant.Mortality в модели Infant Mortality ~ . Catholic\*Agriculture имеет положительный коэффициент, а значит, что в семье крестьянина-фермера из католического региона ниже шанс детской смертности.

Также в задаче 2 были найдены доверительные интервалы для всех коэффициентов в рассматриваемой модели (при p=95%). Было выявлено, что значение коэффициента перед регрессором Argiculture×Education может быть равно 0, а значит данная объясняющая переменная практически не связана с объясняемой переменной Infant.Mortality.

Затем, был построен доверительный интервал для одного запроса со следующими значениями:

1. Catholic – 10 объектов.
2. Agriculture – 20 объектов.
3. Education – 8 объектов.

Был вычислен прогноз модели, который составил 20.16, а также доверительный интервал для рассматриваемой модели. Так, нижняя граница доверительного интервала - 18.34, верхняя граница доверительного интервала – 22.

В задаче 3 были проанализированы данные 14 волны мониторинга экономического положения и здоровья населения РФ. С помощью данных о поле, семейном положении, образовании, возрасте, типе населенного пункта, часов работы, регионе, удовлетворенности трудом, владельце предприятия, информации о имеет ли респондент несколько работ и курит ли респондент, после обработки данных и избавлений от столбцов с большим VIF была построена модель Salary ~ sex + higher\_educ + + status + dur + region + satisfy + foreign\_owner + second\_job + smokes. Данная модель хоть и имеет достаточно значимые переменные, однако ее мал и составляет . Зависимость можно описать следующим образом: в 2004 году наибольшую прибыль мог бы получить мужчина с высшим образованием, который имеет средний возраст и проживает в крупном регионе, а сам он работает на иностранное предприятие. С помощью тех же данных из 14 волны были выделены 2 подвыборки, это не замужние женщины и разведенные женщины, разведенные в городе. Только первая подвыборка имеет значимые переменные модели и из нее видно, что наибольшую зарплату получат женщины, которые имеют высшее образование, проживающие в крупном регионе и работают на заграничное предприятие. Вторая подвыборка практически не имеет значимых регрессоров, а значит, из нее нельзя сделать какой-либо адекватный вывод.

В задаче 4 с помощью данных банковских аккаунтов было построено три классификатора, которые предсказывают, имеет ли клиент банка возраст меньше среднего. Так, первый классификатор — это решающее дерево высотой 4 без подбора параметров предсказывает возраст следующим образом - если человек зарегистрирован в банковской системе был недавно, или же у него зарегистрировано много доходов на иждивении и у него меньше транзакций, то данный человек имеет возраст меньше среднего.

Второй классификатор является решающим деревом с высотой 3 и с подбором гиперпараметров с помощью Grid Search. Данный классификатор нашел более эффективное дерево - если человек был зарегистрирован недавно, или он имеет много доходов на иждивении, то его возраст меньше среднего.

Наконец, третий классификатор является случайным лесом, состоящий из 100 деревьев.

Все классификатора имеют практически одинаковую точность, варьирующуюся от 79 % до 81%, а значит, что разумнее всего для предсказания использовать решающее дерево с подбором гиперпараметров, так как оно одно и имеет наименьшую высоту в 3.

В задаче 5 был обработан заданный датасет экзопланет Open Exoplanet Catalogue. Из датасета были убраны столбцы, в которых отсутствует более 50% наблюдений, в оставшихся столбцах был произведен поиск аномальных значений, которые были либо удалены, либо приведены к более усредненному значению. Затем, был произведен поиск столбцов с категориальным признаком, которые были приведены далее к бинарному признаку. Остальные же столбцы с действительными значениями были нормированы.

С помощью полученного датасета было произведено понижение размерности с помощью PCA до 8 компонент, а DBSCAN на t-SNE обнаружил 5 различных типов планет.

В задаче 6 полученный в 5 задаче датасет был снова обработан, но уже без столбцов наклона планеты, а также с помощью метода главных компонент размерность данных была понижена до 3. После PCA был применен метод ближайших соседей, с помощью которого было получено 10 различных кластеров. Каждый кластер содержит в себе планеты, различающиеся по размеру экзопланеты, ее отдаленности от солнца и типу наиближайшей звезды. Наиболее же интересным кластером является кластер №7, в который попали не только меркурии, но также и потенциальные земли. Так, помимо самой Земли в данном кластере находится одна из действительно потенциальных и оптимистичных на жизнь планета HD 40307 g.

Оценив всю работу в целом, можно сделать следующие выводы:

* + 1. Не смотря на свою простоту в использовании, линейная регрессия является мощным инструментом для анализа зависимостей в данных. Так, были решены задачи при поиске зависимостей между уровнем образования человека между его социальным классом и наличием у него детей, а также был сделан поиск зависимостей между детской смертностью и религиозностью в регионе и сельским хозяйством. С помощью той же линейной регрессии мы смогли предсказать зарплату гражданина причем узнав, что мужчины, работающие в крупном регионе на иностранное предприятие, по данным опроса 2004 года, зарабатывают намного больше, чем остальные.
    2. Классификация является таким же необходимым инструментом для работы с данными. Так, используя решающее дерево и случайны лес мы смогли обнаружить, что, зная только дату регистрации банковского аккаунта и количества дохода на иждивении с 80% точностью можно предсказать возраст человека.
    3. С помощью понижения размерности данных и кластеризации можно разделить выборку даже если она была неразмеченной, причем при более внимательном рассмотрении полученных данных можно обнаружить новые зависимости. Так, в данной работе после обработки датасета экзопланет и с последующим понижением до 3 компонент методом PCA, после кластеризации методом ближайших соседей было обнаружено 10 кластеров, экзопланеты в которых отличаются от других. Причем в одном из кластеров расположена и сама Земля, а также были найдены другие кандидаты на планеты пригодные для жизни.

А значит, посредством построения зависимостей различных переменных друг от друга становится возможным получить более широкий и понятный взгляд на происходящее в данных. Для этого можно использовать как линейную регрессию, как классификацию, так и кластеризацию с понижением размерности в зависимости от поставленной задачи, причем полученную модель при ее удачном обучении можно использовать в будущем на новых данных.

Список литературы

1. У. Маккинни. Python и анализ данных [Текст] / У. Маккинни – Москва: ДМК Пресс, 2015. – 481 с.
2. Scikit Learn User Guide [Электронный ресурс] / Режим доступа: https://scikit-learn.org/stable/user\_guide.html, свободный.
3. R Document Collections, Journals and Proceedings [Электронный ресурс] / Режим доступа: https://www.r-project.org/other-docs.html, свободный.
4. Pandas Documentation [Электронный ресурс] / Режим доступа: https://pandas.pydata.org/docs/, свободный.
5. Бреннан П.Х. Life in Our Solar System? Meet the Neighbors [Электронный ресурс] / Бреннан П.Х – Режим доступа: https://exoplanets.nasa.gov/news/1665/life-in-our-solar-system-meet-the-neighbors/, свободный.

Приложение 1

#Код задачи

require(stats); require(graphics)

plot(swiss)

head(swiss)

sprintf("Среднее значение: %.2f", mean(swiss$Education))

sprintf("Дисперсия: %.2f", var(swiss$Education))

sprintf("Среднеквадратическое отклонение: %.2f", sd(swiss$Education))

sprintf("Среднее значение: %.2f", mean(swiss$Agriculture))

sprintf("Дисперсия: %.2f", var(swiss$Agriculture))

sprintf("Среднеквадратическое отклонение: %.2f", sd(swiss$Agriculture)) sprintf("Среднее значение: %.2f", mean(swiss$Fertility))

sprintf("Дисперсия: %.2f", var(swiss$Fertility))

sprintf("Среднеквадратическое отклонение: %.2f", sd(swiss$Fertility))

install.packages("lmtest", dependencies=TRUE)

library("lmtest")

model = lm(Education~Fertility, swiss)

summary(model)

model = lm(Education~Agriculture, swiss)

summary(model)

Приложение 2.1

ca\_ar = lm(Catholic~Agriculture, swiss)

summary(ca\_ar)

ca\_ed = lm(Catholic~Education, swiss)

summary(ca\_ed)

ag\_ed = lm(Agriculture~Education, swiss)

summary(ca\_ed)

#2

model = lm(Infant.Mortality ~ Catholic + Agriculture + Education , swiss)

summary(model)

# 3

model\_log = lm(Infant.Mortality ~ I(log(Catholic)) + I(log(Agriculture)) + I(log(Education)) , swiss)

summary(model\_log)

model\_log = lm(Infant.Mortality ~ Catholic + Agriculture + Education + I(log(Catholic)) + I(log(Agriculture)) + I(log(Education)) , swiss)

summary(model\_log)

# 4

model\_prod1 = lm(Infant.Mortality ~ Catholic + Agriculture + Education + I(Catholic^2) + I(Agriculture^2)

+ I(Education^2) + I(log(Catholic)) + I(log(Agriculture)) + I(log(Education)) + I(Catholic\*Agriculture)

+ I(Catholic\*Education) + I(Agriculture\*Education), swiss)

summary(model\_prod1)

vif(model\_prod1)

model\_prod1 = lm(Infant.Mortality ~  Agriculture + Education + I(Catholic^2) + I(Agriculture^2)

+ I(Education^2) + I(log(Catholic)) + I(log(Agriculture)) + I(log(Education)) + I(Catholic\*Agriculture)

+ I(Catholic\*Education) + I(Agriculture\*Education), swiss)

summary(model\_prod1)

vif(model\_prod1)

model\_prod1 = lm(Infant.Mortality ~ Education + I(Catholic^2) + I(Agriculture^2)

+ I(Education^2) + I(log(Catholic)) + I(log(Agriculture)) + I(log(Education)) + I(Catholic\*Agriculture)

+ I(Catholic\*Education) + I(Agriculture\*Education), swiss)

summary(model\_prod1)

vif(model\_prod1)

model\_prod1 = lm(Infant.Mortality ~ I(Catholic^2) + I(Agriculture^2)

+ I(Education^2) + I(log(Catholic)) + I(log(Agriculture)) + I(log(Education)) + I(Catholic\*Agriculture)

+ I(Catholic\*Education) + I(Agriculture\*Education), swiss)

summary(model\_prod1)

model\_prod1 = lm(Infant.Mortality ~ I(Catholic^2) + I(Agriculture^2)

+ I(Education^2) + I(log(Catholic)) + I(Catholic\*Agriculture)

 + I(Agriculture\*Education), swiss)

summary(model\_prod1)

vif(model\_prod1)

model\_prod1 = lm(Infant.Mortality ~ I(Catholic^2) + I(Agriculture^2)

+ I(Education^2) + I(log(Catholic)) + I(Catholic\*Agriculture)

 + I(Agriculture\*Education), swiss)

summary(model\_prod1)

Приложение 2.2

library("car")

require(stats)

require(graphics)

plot(swiss)

model = lm(Infant.Mortality ~ I(Catholic^2) + I(Agriculture^2) + I(Education^2) + I(log(Catholic)) + I(Catholic\*Agriculture) + I(Agriculture\*Education), swiss)

summary(model)

qt(0.95, df = 33)

data\_new = data.frame(Catholic = 10, Agriculture= 20, Education= 8)

predict(model, data\_new, interval = "confidence")

Приложение 3

install.packages("devtools")

devtools::install\_github("bdemeshev/rlms")

library("lmtest")

library("rlms")

library("dplyr")

library("GGally")

library("car")

library("sandwich")

"""

jj13.2 - зп ! check

jh5 - пол ! check

j\_marst - семейное положение ! check

j\_educ - ВО ! check

j\_age - возраст ! check

status - тип населенного пункта ! check

jj6.2 - Часов работы ! check

psu - регион check

jj1.1.1 - удовлетворенность трудом check

jj24 - являются ли иностранные фирмы владельцем check

jj32.1 - несколько работ check

jm71 - курит ли человек check

"""

o\_data <- rlms\_read("C:/Users/ruthh/Documents/workspace/data/r14iall\_32.sav")

glimpse(o\_data)

data = select(o\_data, jj13.2, jh5, j\_educ, j\_age, status, jj6.2, psu, jj1.1.2, jj24, jj32, jm71, j\_marst)

data = na.omit(data)

data

#зарплата c элементами нормализации

sal = as.numeric(data$jj13.2)

sal1 = as.character(data$jj13.2)

sal2 = lapply(sal1, as.integer)

sal = as.numeric(unlist(sal2))

mean(sal)

data["salary"] = (sal - mean(sal)) / sqrt(var(sal))

data["salary"] = data["salary"]\*1.034

#пол

data["sex"]=data$jh5

data$sex[which(data$sex!=1)] <- 0

data$sex[which(data$sex==1)] <- 1

#семейное положение

data["wed"]= data$j\_marst

data$wed1 = 0

data$wed1[which(data$wed==1)] <- 1

data$wed1[which(data$wed==3)] <- 1

data$wed2 = 0

data$wed2[which(data$wed==2)] <- 1

data["wed3"]=data$j\_marst

data$wed3 = 0

data$wed3[which(data$wed==4)] <- 1

data["wed4"]=data$j\_marst

data$wed4 = 0

data$wed4[which(data$wed==5)] <- 1

#образование

data["h\_educ"] = data$j\_educ

data["higher\_educ"] = 0

data$higher\_educ[which(data$h\_educ==21)] <- 1

data$higher\_educ[which(data$h\_educ==22)] <- 1

data$higher\_educ[which(data$h\_educ==23)] <- 1

#возраст c элементами нормализации

age = as.character(data$j\_age)

age = lapply(age, as.integer)

age = as.numeric(unlist(age))

data["age"]= (age - mean(age)) / sqrt(var(age))

data["age"]

#населенный пункт

data["status1"]=data$status

data["status2"] = 0

data$status2[which(data$status1==1)] <- 1

data$status2[which(data$status1==2)] <- 1

data["status"] = data$status2

#продолжительность рабочей недели

dur1 = as.character(data$jj6.2)

dur2 = lapply(dur1, as.integer)

dur3 = as.numeric(unlist(dur2))

data["dur"] = (dur3 - mean(dur3)) / sqrt(var(dur3))

########## ДАЛЕЕ ИДУТ ВЫБРАННЫЕ СТОЛБЦЫ

#region

data["region"]=0

data$region[which(data$psu==1)] <- 1

data$region[which(data$psu==2)] <- 1

data$region[which(data$psu==3)] <- 1

data$region[which(data$psu==16)] <- 1

data$region[which(data$psu==32)] <- 1

#удовлетворенность

data["sat"]=data$jj1.1.2

data["satisfy"] = 0

data$satisfy[which(data$sat==1)] <- 1

data$satisfy[which(data$sat==2)] <- 1

#другое государство - совладелец предприятия

data["foreign"] = data$jj24

data["foreign\_owner"] = 0

data$foreign\_owner[which(data$foreign==1)] <- 1

#есть приработок, вторая, n-я работа

data["sj"] = data$jj32

data["second\_job"] = 0

data$second\_job[which(data$sj==1)] <- 1

#Курит ли человек

data["sp"] = data$jm71

data["smokes"] = 0

data$smokes[which(data$sp==1)] <- 1

data = na.omit(data)

data = select(data, salary, sex, wed1,wed2,wed3,wed4, higher\_educ, age, status, dur, region,

satisfy, foreign\_owner, second\_job, smokes)

#среднее, медиана и мода для каждого параметра

uniqv <- unique(data$salary)

u1 = uniqv[which.max(tabulate(match(data$salary, uniqv)))]

table\_salary = data.frame(Value = "salary", Mean = mean(data$salary,na.rm = TRUE), Median = median(data$salary,na.rm = TRUE), Mode = u1)

table\_salary

mean(data$sex)

mean(data$wed1)

mean(data$wed2)

mean(data$wed3)

mean(data$wed4)

mean(data$higher\_educ)

uniqv <- unique(data$age)

u1 = uniqv[which.max(tabulate(match(data$age, uniqv)))]

table\_age = data.frame(Value = "age", Mean = mean(data$age,na.rm = TRUE), Median = median(data$age,na.rm = TRUE), Mode = u1)

table\_age

mean(data$status)

uniqv <- unique(data$dur)

u1 = uniqv[which.max(tabulate(match(data$dur, uniqv)))]

table\_dur = data.frame(Value = "dur", Mean = mean(data$dur,na.rm = TRUE), Median = median(data$dur,na.rm = TRUE), Mode = u1)

table\_dur

mean(data$region)

mean(data$satisfy)

mean(data$foreign\_owner)

mean(data$second\_job)

mean(data$smokes)

#графики парных зависимостей

ggpairs(data)

#уровни факторных переменных

qplot(data = data, salary)

qplot(data = data, age)

qplot(data = data, dur)

#построение зависимостей

model = lm(data = data, salary ~ sex + wed1 + wed2 + wed3 + higher\_educ + age + status + dur + region +

satisfy + foreign\_owner + second\_job + smokes)

summary(model)

vif(model)

# wed1 wed2 и wed3 имеют очень большой vif, как и p-значение - они не важны.

model2 = lm(data = data, salary ~ sex + higher\_educ + age + status + dur + region +

satisfy + foreign\_owner + second\_job + smokes)

summary(model2)

vif(model2)

waldtest(model2)

waldtest(model)

waldtest(model2, model)

bptest(model2)

gqtest(model2, order.by = ~salary, data = data, fraction = 0.2)

#наблюдается гетероскедастичность по тесту Бройша-Паганаи и по Голдфельду-Квандту.

#оценка коэффициентов и их ошибки. Доверительные интервалы: коэффициент +- 1.96\*ошибка

conftable = coeftest(model1, vcov.=vcovHC(model2))

conftable[,1]

conftable[,2]

#Введем нелинейные регрессоры

# Найдем наилучшую модель с помощью перебора

best\_model = model2

summary(best\_model)

# Выделим следующие случаи:

# i. степень для age

for(i in seq(0.1, 2, 0.1)) {

#print(i)

i\_model = lm(data = data, salary ~ sex + higher\_educ + age + I(age^i) + status + dur + region +

satisfy + foreign\_owner + second\_job + smokes)

if(summary(i\_model)$r.squared > summary(best\_model)$r.squared){

best\_model = i\_model

print(i)

print("стал лучшим коэффициентом для age")

}

}

summary(best\_model)

# ii. степень для dur

for(i in seq(0.1, 2, 0.1)) {

#print(i)

i\_model = lm(data = data, salary ~ sex + higher\_educ + age + status + dur + I(dur^i) + region +

satisfy + foreign\_owner + second\_job + smokes)

if(summary(i\_model)$r.squared > summary(best\_model)$r.squared){

best\_model = i\_model

print(i)

print("стал лучшим коэффициентом для dur")

}

}

summary(best\_model)

# iii. степень для salary

for(i in seq(0.1, 2, 0.1)) {

#print(i)

i\_model = lm(data = data, salary^i ~ sex + higher\_educ + age + status + dur + region +

satisfy + foreign\_owner + second\_job + smokes)

if(summary(i\_model)$r.squared > summary(best\_model)$r.squared){

best\_model = i\_model

print(i)

print("стал лучшим коэффициентом для salary")

}

}

# iv. age и dur

for(i in seq(0.1, 2, 0.1)) {

#print(i)

i\_model = lm(data = data, salary ~ sex + higher\_educ + age + I(age^i) + status + dur + I(dur^i) + region +

satisfy + foreign\_owner + second\_job + smokes)

#print(summary(i\_model)$r.squared)

if(summary(i\_model)$r.squared > summary(best\_model)$r.squared){

best\_model = i\_model

print(i)

print("стал лучшим коэффициентом для age and dur")

}

}

summary(best\_model)

# v. salary и dur

for(i in seq(0.1, 2, 0.1)) {

#print(i)

i\_model = lm(data = data, salary^i ~ sex + higher\_educ + age + status + dur + I(dur^i) + region +

satisfy + foreign\_owner + second\_job + smokes)

#print(summary(i\_model)$r.squared)

if(summary(i\_model)$r.squared > summary(best\_model)$r.squared){

best\_model = i\_model

print(i)

print("стал лучшим коэффициентом для age and dur")

}

}

# vi. salary и age

for(i in seq(0.1, 2, 0.1)) {

#print(i)

i\_model = lm(data = data, salary^i ~ sex + higher\_educ + age + I(age^i) + status + dur + region +

satisfy + foreign\_owner + second\_job + smokes)

#print(summary(i\_model)$r.squared)

if(summary(i\_model)$r.squared > summary(best\_model)$r.squared){

best\_model = i\_model

print(i)

print("стал лучшим коэффициентом для age and dur")

}

}

# vii. все

for(i in seq(0.1, 2, 0.1)) {

#print(i)

i\_model = lm(data = data, salary^i ~ sex + higher\_educ + age + I(age^i) + status + dur + I(age^i) + region +

satisfy + foreign\_owner + second\_job + smokes)

#print(summary(i\_model)$r.squared)

if(summary(i\_model)$r.squared > summary(best\_model)$r.squared){

best\_model = i\_model

print(i)

print("стал лучшим коэффициентом для age and dur")

}

}

# Проверим то же для логарифма

# i. логарифм для age

for(i in seq(0.1, 2, 0.1)) {

#print(i)

i\_model = lm(data = data, salary ~ sex + higher\_educ + age + I(log(i, base=age)) + status + dur + region +

satisfy + foreign\_owner + second\_job + smokes)

if(summary(i\_model)$r.squared > summary(best\_model)$r.squared){

best\_model = i\_model

print(i)

print("стал лучшим коэффициентом для age")

}

}

summary(best\_model)

# ii. степень для dur

for(i in seq(0.1, 2, 0.1)) {

#print(i)

i\_model = lm(data = data, salary ~ sex + higher\_educ + age + status + dur + I(log(i, dur)) + region +

satisfy + foreign\_owner + second\_job + smokes)

if(summary(i\_model)$r.squared > summary(best\_model)$r.squared){

best\_model = i\_model

print(i)

print("стал лучшим коэффициентом для dur")

}

}

summary(best\_model)

# iii. age и dur

for(i in seq(0.1, 2, 0.1)) {

#print(i)

i\_model = lm(data = data, salary ~ sex + higher\_educ + age + I(log(i, age)) + status + dur + I(log(i, dur)) + region +

satisfy + foreign\_owner + second\_job + smokes)

#print(summary(i\_model)$r.squared)

if(summary(i\_model)$r.squared > summary(best\_model)$r.squared){

best\_model = i\_model

print(i)

print("стал лучшим коэффициентом для age and dur")

}

}

# Как итог, самой лучшей моделью стала модель с нелинейным коэффициентом age^2

# Попробуем дополнительно ввести логарифмы

summary(best\_model)

for(i in seq(0.1, 2, 0.1)) {

#print(i)

i\_model = lm(data = data, salary ~ sex + higher\_educ + age + I(age^i) + status + dur + region +

satisfy + foreign\_owner + second\_job + smokes)

if(summary(i\_model)$r.squared > summary(best\_model)$r.squared){

best\_model = i\_model

print(i)

print("стал лучшим коэффициентом для log age")

}

}

for(i in seq(0.1, 2, 0.1)) {

#print(i)

i\_model = lm(data = data, salary ~ sex + higher\_educ + age + I(age^i) + status + dur + I(log(i, dur)) + region +

satisfy + foreign\_owner + second\_job + smokes)

if(summary(i\_model)$r.squared > summary(best\_model)$r.squared){

best\_model = i\_model

print(i)

print("стал лучшим коэффициентом для age")

}

}

model3 = lm(data = data, salary ~ sex + higher\_educ + age + I(age^2) + status + dur + I(dur^2) + region +

satisfy + foreign\_owner + second\_job + smokes)

summary(model3)

vif(model3)

# У age крайне малый p. Посмотрим что будет, если удалить его

model3 = lm(data = data, salary ~ sex + higher\_educ + I(age^2) + status + dur + I(dur^2) + region +

satisfy + foreign\_owner + second\_job + smokes)

# Сделаем то же самое для dur^2

model3 = lm(data = data, salary ~ sex + higher\_educ + I(age^2) + status + dur + region +

satisfy + foreign\_owner + second\_job + smokes)

summary(model3)

vif(model3)

# R^2 изменился не критично, а вздутие уменьшилось. Можно оставить в таком виде

# Судя по обученным коэффициентам модели, можно сделать следующие выводы:

# 1. Пол, к сожалению, влияет на зарплату опрошенных сотрудников. Мужчина получает больше, чем женщина.

# 2. Наличие высшего образования влияет на зарплату в некоторой мере.

# 3. Возраст отрицательно влияет на зп - чем старше или моложе человек относительно среднего (для предоставленных данных) возраста, тем больше шанс, что платить будут меньше

# 4. Статус населенного пункта сказывается на зп

# 5. Чем больше человек работает, тем больше (но не сильно больше) он получит денег

# 6. Регион определеленно влияет на прибыль - в крупных регионах получают больше

# 7. Если человек доволен работой, значит, скорее всего, он получает достатучную прибыль

# 8. Иностранное предприятие принесет большую прибыль

# 9. Если у человека несколько работ, значит, скорее всего, он получает недостатучную прибыль, из-за чего ему и нужно несколько работ

# 10. Курение не зависит на прибыль. Возможно, сигареты тогда стоили меньше, и не могли повлиять на бюджет

# Значит, индивид, который:

# Имеет мужской пол

# Имеет высшее образование

# Среднего (из предоставленных данных) возраста

# Из крупного региона

# Работающий на иностранное предприятие

# Получит наибольшую прибыль.

data

# Оценим полученную модель на выборке Разведённыъ незамужних женщин (), которые живущит в городе.

data <-filter(data, sex==0)

data

data\_a <-filter(data, data$wed1==1 | data$wed3==1 | data$wed4==1)

data\_a

data\_b <- filter(data, data$status==1 & data$wed3==1)

data\_b

model\_a = lm(data = data\_a, salary ~ higher\_educ + I(age^2) + status + dur + region +

satisfy + foreign\_owner + second\_job + smokes)

summary(model\_a)

# R^2 не сильно изменился, но p-значение значительно ухудшилось - status, durability и satisfy объясняют хуже

# наибольшую зп получат женщины, которые имеют высшее образование, из крупного региона, работают на заграничное предприятие.

model\_b = lm(data = data\_b, salary ~ higher\_educ + I(age^2) + dur + region +

satisfy + foreign\_owner + second\_job + smokes)

summary(model\_b)

# R^2 стал лучше, но p-значение осталось хорошим только у region и foreign owner

# наибольшую зп получат женщины, которые проживают в крупном регионе и работают на иностранное предприятие.

Приложение 4

import pandas

import numpy as np

from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import cross\_val\_score

from sklearn.model\_selection import GridSearchCV

data = pandas.read\_csv('BankChurners.csv')

data = data.dropna()

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

pandas.unique(data["Gender"].values)

pandas.unique(data["Attrition\_Flag"].values)

pandas.unique(data["Education\_Level"].values)

pandas.unique(data["Marital\_Status"].values)

pandas.unique(data["Income\_Category"].values)

pandas.unique(data["Card\_Category"].values)

data['Gender'] = np.where(data['Gender'] == 'M', 0, 1)

data['Attrition\_Flag'] = np.where(data['Attrition\_Flag'] == 'Existing Customer', 0, 1)

def conditions(x):

if x == 'High School':

return 1

elif x == 'College':

return 1

elif x == 'Doctorate':

return 1

else:

return 0

func = np.vectorize(conditions)

data['Education\_Level'] = func(data['Education\_Level'])

data['Marital\_Status'] = np.where(data['Marital\_Status'] == 'Married', 0, 1)

def conditions(x):

if x == 'Less than $40K':

return 0

elif x == 'Unknown':

return 0

else:

return 1

func = np.vectorize(conditions)

data['Income\_Category'] = func(data['Income\_Category'])

data['Card\_Category'] = np.where(data['Card\_Category'] == 'Blue', 1, 0)

data.head()

ages = data["Customer\_Age"]

data["Customer\_Age"] = (ages - ages.mean()) / np.sqrt(ages)

month = data["Months\_on\_book"]

data["Months\_on\_book"] = (month - month.mean()) / np.sqrt(month)

month = data["Total\_Revolving\_Bal"]

data["Total\_Revolving\_Bal"] = (month - month.mean()) / np.sqrt(month)

month = data["Avg\_Open\_To\_Buy"]

data["Avg\_Open\_To\_Buy"] = (month - month.mean()) / np.sqrt(month)

month = data["Total\_Amt\_Chng\_Q4\_Q1"]

data["Total\_Amt\_Chng\_Q4\_Q1"] = (month - month.mean()) / np.sqrt(month)

month = data["Total\_Trans\_Amt"]

data["Total\_Trans\_Amt"] = (month - month.mean()) / np.sqrt(month)

month = data["Total\_Trans\_Ct"]

data["Total\_Trans\_Ct"] = (month - month.mean()) / np.sqrt(month)

month = data["Total\_Ct\_Chng\_Q4\_Q1"]

data["Total\_Ct\_Chng\_Q4\_Q1"] = (month - month.mean()) / np.sqrt(month)

month = data["Credit\_Limit"]

data["Credit\_Limit"] = (month - month.mean()) / np.sqrt(month)

data.replace([np.inf, -np.inf], np.nan, inplace=True)

data.dropna()

data = data.iloc[:, :-2]

X = data

def conditions(x):

    if x <= 0:

        return 1

    else:

        return 0

func = np.vectorize(conditions)

X['Customer\_Age'] = func(X['Customer\_Age'])

Y = X["Customer\_Age"]

X = X.drop('Customer\_Age', 1)

X.fillna(X.mean(), inplace=True)

Y.fillna(Y.mean(), inplace=True)

x\_train, x\_validation, y\_train, y\_validation = train\_test\_split(X, Y, test\_size=.33, random\_state=1)

T = DecisionTreeClassifier(random\_state=42, max\_depth = 4)

T = T.fit(x\_train, y\_train)

from sklearn.tree import plot\_tree

from matplotlib import pyplot as plt

fig = plt.figure(figsize=(25,25))

\_ = plot\_tree(T, filled=True, class\_names=True)

print("accuracy:", np.average(cross\_val\_score(T, x\_validation, y\_validation, scoring='accuracy')))

print("f1:", np.average(cross\_val\_score(T, x\_validation, y\_validation, scoring='f1')))

print("precision:", np.average(cross\_val\_score(T, x\_validation, y\_validation, scoring='precision')))

print("recall:", np.average(cross\_val\_score(T, x\_validation, y\_validation, scoring='recall')))

import warnings

warnings.filterwarnings('ignore')

tree = DecisionTreeClassifier(class\_weight=None, criterion='gini', max\_depth=None,

            max\_features=None, max\_leaf\_nodes=None,

            min\_impurity\_decrease=0, min\_samples\_leaf=2, min\_samples\_split=3,

            min\_weight\_fraction\_leaf=0, presort=False, random\_state=42,

            splitter='best')

params = {

    'max\_depth': list(range(2, 30)),

    'min\_samples\_split': [2, 3, 4, 5, 6, 7, 8, 9, 10]

}

gridsearch = GridSearchCV(cv=3, error\_score='raise-deprecating',

       estimator=tree, iid='warn', n\_jobs=-1,

       param\_grid=params, pre\_dispatch='2\*n\_jobs',

       refit=True, return\_train\_score='warn',

       verbose=1)

gridsearch.fit(x\_train, y\_train)

T = gridsearch.best\_estimator\_

T

print("accuracy:", np.average(cross\_val\_score(T, x\_validation, y\_validation, scoring='accuracy')))

print("f1:", np.average(cross\_val\_score(T, x\_validation, y\_validation, scoring='f1')))

print("precision:", np.average(cross\_val\_score(T, x\_validation, y\_validation, scoring='precision')))

print("recall:", np.average(cross\_val\_score(T, x\_validation, y\_validation, scoring='recall')))

fig = plt.figure(figsize=(25,25))

\_ = plot\_tree(T, filled=True, class\_names=True)

print("X[7]:", X.columns[7])

print("X[2]:", X.columns[2])

from sklearn.ensemble import RandomForestClassifier

from sklearn.model\_selection import GridSearchCV

param\_grid = { 'n\_estimators': [5, 10, 50, 100, 200, 300, 400],'max\_features': ['auto'],'max\_depth' : list(range(2, 20)), 'criterion' :['gini']}

RFC = GridSearchCV(estimator=RandomForestClassifier(), param\_grid=param\_grid, cv= 5, refit = True)

RFC.fit(x\_train, y\_train)

print("accuracy:"+str(np.average(cross\_val\_score(RFC.best\_estimator\_, x\_validation, y\_validation, scoring='accuracy'))))

print("f1:"+str(np.average(cross\_val\_score(RFC.best\_estimator\_, x\_validation, y\_validation, scoring='f1'))))

print("precision:"+str(np.average(cross\_val\_score(RFC.best\_estimator\_, x\_validation, y\_validation, scoring='precision'))))

print("recall:"+str(np.average(cross\_val\_score(RFC.best\_estimator\_, x\_validation, y\_validation, scoring='recall'))))

len(RFC.best\_estimator\_.estimators\_)

RFC.best\_estimator\_.estimators\_

Приложение 5.1

# -\*- 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 5.2

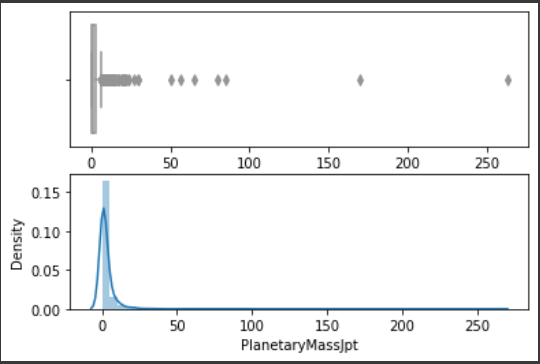


Рисунок 1. Ящик с усами для массы планеты.

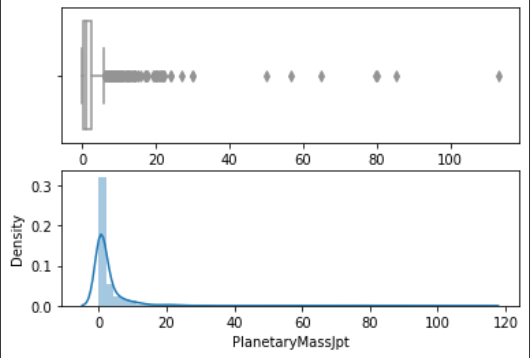


Рисунок 2. Ящик с усами для массы планеты после редактирования выбросов и аномалий.

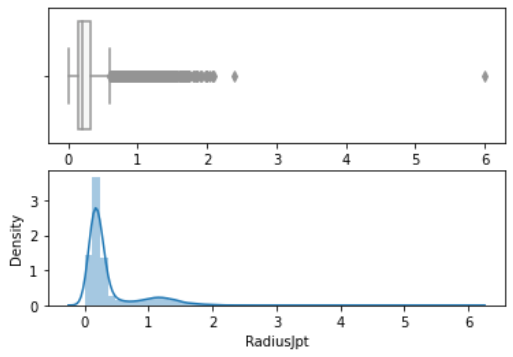


Рисунок 3. Ящик с усами для радиуса планеты.

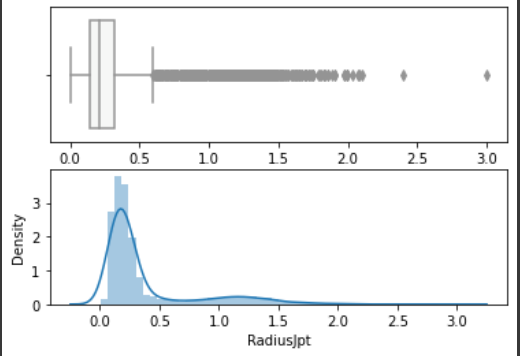


Рисунок 4. Ящик с усами для радиуса планеты после редактирования выбросов и аномалий.

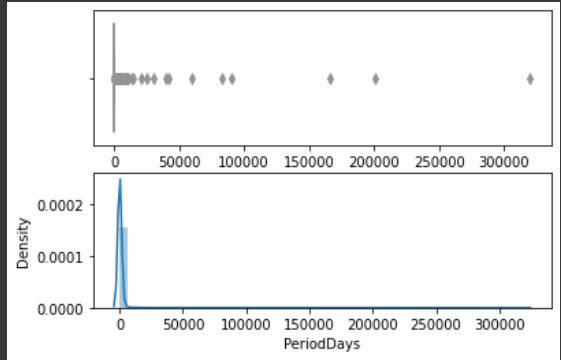


Рисунок 5. Ящик с усами для периода планеты.

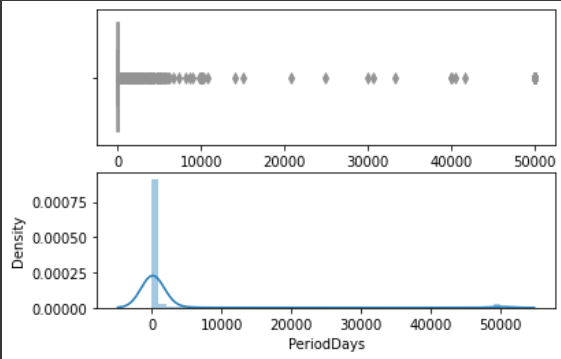


Рисунок 6. Ящик с усами для периода планеты после редактирования выбросов и аномалий.

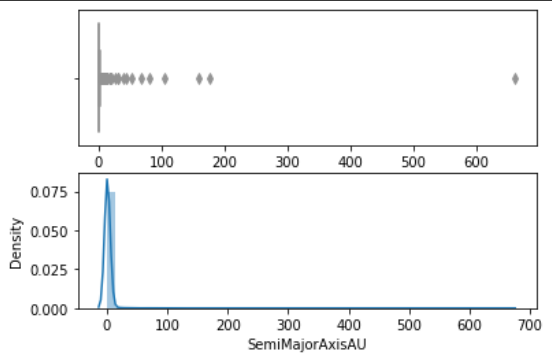


Рисунок 7. Ящик с усами для наклона планеты.

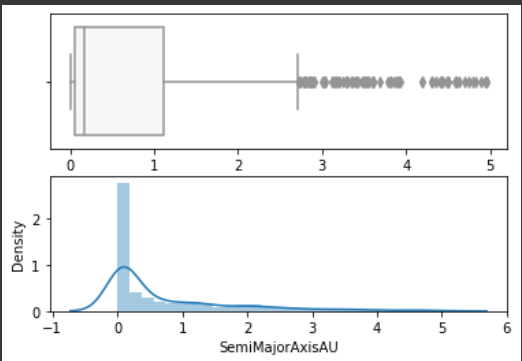


Рисунок 8. Ящик с усами для наклона планеты после редактирования выбросов и аномалий.

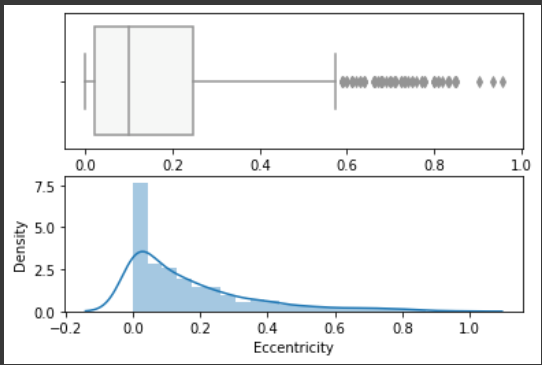


Рисунок 9. Ящик с усами для эксцентриситета планеты.

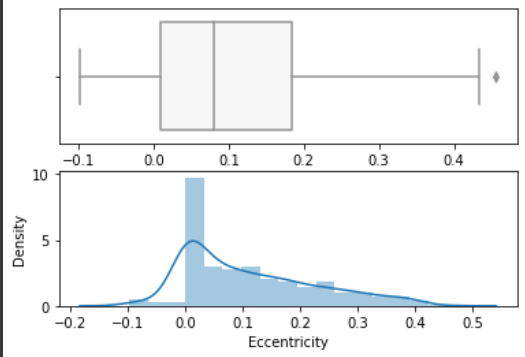


Рисунок 10. Ящик с усами для эксцентриситета планеты после редактирования выбросов и аномалий.

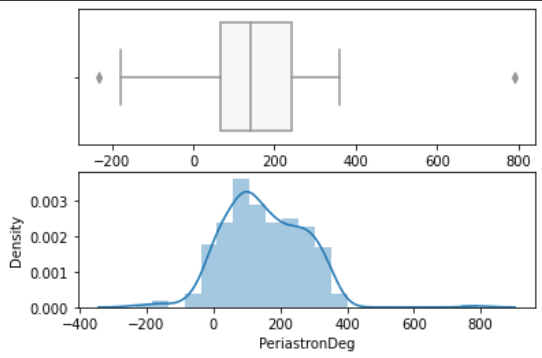


Рисунок 11. Ящик с усами для наклона периастра планеты.

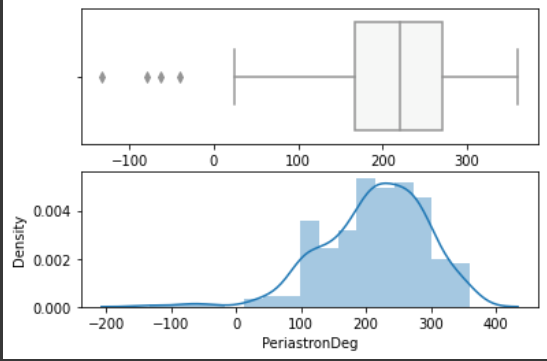


Рисунок 12. Ящик с усами для наклона периастра планеты после редактирования выбросов и аномалий.

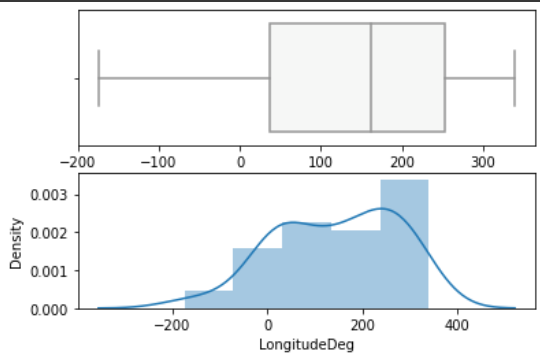


Рисунок 13. Ящик с усами для долготы планеты.

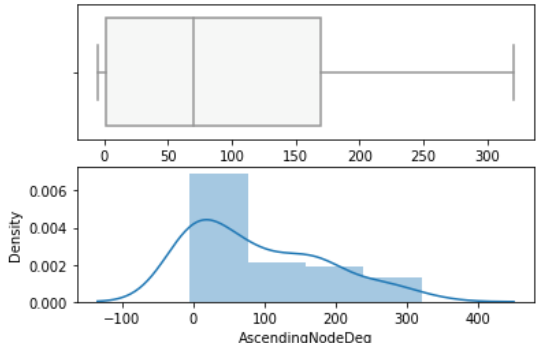


Рисунок 14. Ящик с усами для восходящего узла планеты.

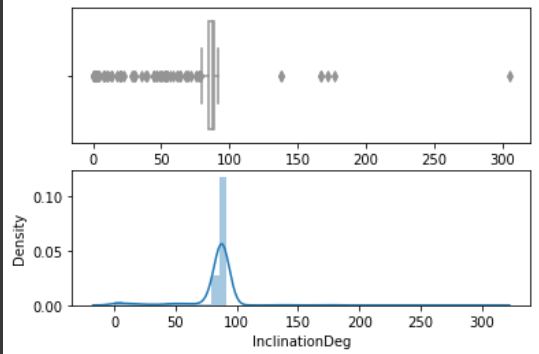


Рисунок 14. Ящик с усами для наклона планеты.

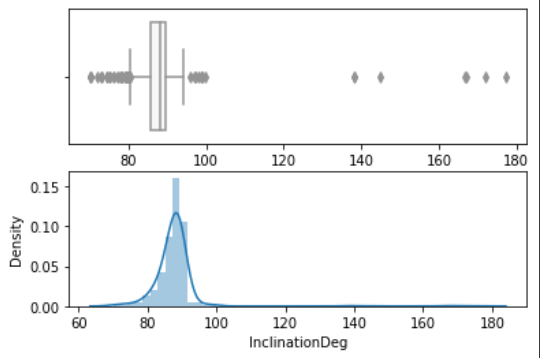


Рисунок 15. Ящик с усами для наклона планеты после редактирования выбросов и аномалий.

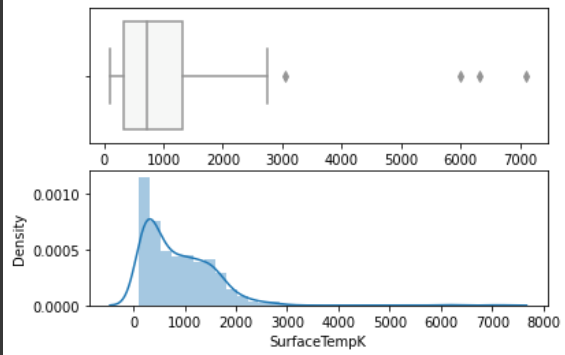


Рисунок 16. Ящик с усами для температуры планеты.

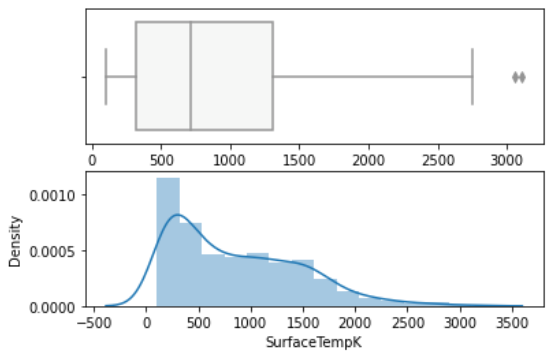


Рисунок 17. Ящик с усами для температуры планеты после редактирования выбросов и аномалий.

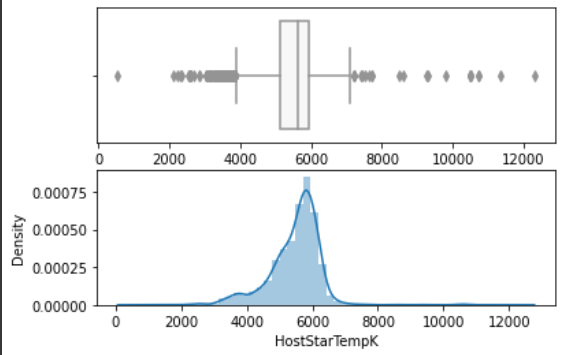


Рисунок 18. Ящик с усами для температуры звезды.

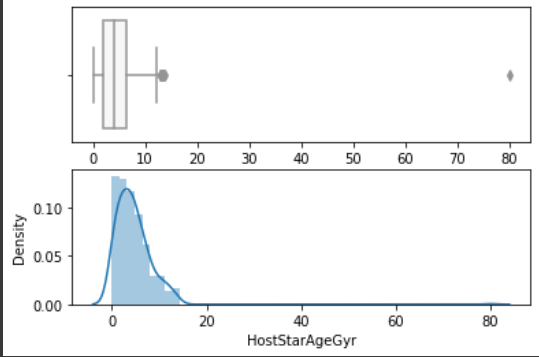


Рисунок 19. Ящик с усами для температуры звезды после редактирования выбросов и аномалий.

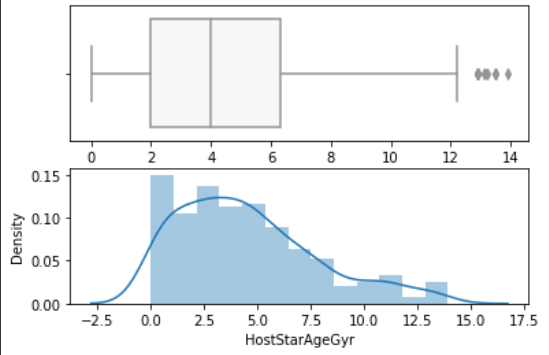


Рисунок 18. Ящик с усами для возраста звезды.

Приложение 6

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

"""6.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1c-SpGu\_YOtY6x9l4XBghgExAb\_fKxmps

"""

!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

from sklearn.decomposition import PCA

from sklearn.manifold import TSNE

from sklearn.cluster import KMeans

from sklearn.cluster import DBSCAN

from pandas\_profiling import ProfileReport

from google.colab import drive

drive.mount('/content/drive')

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

"""# I. Обработка данных"""

mdata = data

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

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

data = mdata

mdata = data

mdata[mdata.RadiusJpt > 2]

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

data = mdata

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

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

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

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

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

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

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

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

data.loc[data.DistFromSunParsec > 2000, 'DistFromSunParsec'] = data.loc[data.DistFromSunParsec > 2000, 'DistFromSunParsec'] - 2000

data.loc[data.DistFromSunParsec > 4000, 'DistFromSunParsec'] = data.loc[data.DistFromSunParsec > 4000, 'DistFromSunParsec'] - 3000

data.loc[data.DistFromSunParsec > 2000, 'DistFromSunParsec'] = data.loc[data.DistFromSunParsec > 2000, 'DistFromSunParsec'] - 2000

data.loc[data.DistFromSunParsec > 1500, 'DistFromSunParsec'] = data.loc[data.DistFromSunParsec > 1500, 'DistFromSunParsec'] - 1500

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

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

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

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 = data.drop('DiscoveryYear', 1)

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

origdata = data

"""# II. Кластеризация со столбцами долготы"""

dolgdata = origdata

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

dolgdata["Dec1"] = np.NaN

dolgdata["Dec2"] = np.NaN

dolgdata["Dec3"] = np.NaN

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

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

if x != x:

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

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

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

else:

x = x.split(' ')

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

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

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

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

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

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

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

dolgdata["Ra1"] = np.NaN

dolgdata["Ra2"] = np.NaN

dolgdata["Ra3"] = np.NaN

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

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

if x != x:

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

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

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

else:

x = x.split(' ')

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

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

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

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

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

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

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

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

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

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

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

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

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

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

dolgdata.RadiusJpt = (dolgdata.RadiusJpt - dolgdata.RadiusJpt.mean()) / dolgdata.RadiusJpt.std()

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

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

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

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

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

dolgdata.Ra2 = (dolgdata.Ra2 - dolgdata.Ra1.mean()) / dolgdata.Ra2.std()

dolgdata.Ra3 = (dolgdata.Ra3 - dolgdata.Ra1.mean()) / dolgdata.Ra3.std()

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

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

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

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

dolgdata.Ra2 = (dolgdata.Ra2 - dolgdata.Ra1.mean()) / dolgdata.Ra2.std()

dolgdata.Ra3 = (dolgdata.Ra3 - dolgdata.Ra1.mean()) / dolgdata.Ra3.std()

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

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

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

pandas\_profiling.ProfileReport(dolgdata)

"""## Снижение размерности"""

pca = PCA()

pca.fit(dolgdata)

X\_pca = pca.transform(dolgdata)

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, dolgdata.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(8, c='b')

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

plt.xlabel('Количество признаков')

plt.ylabel('Описание дисперсии')

"""# III. Кластеризация без столбцов долготы"""

data2 = origdata

data2.PeriodDays = (data2.PeriodDays - data2.PeriodDays.mean()) / data2.PeriodDays.std()

data2.DistFromSunParsec = (data2.DistFromSunParsec - data2.DistFromSunParsec.mean()) / data2.DistFromSunParsec.std()

data2.HostStarRadiusSlrRad = (data2.HostStarRadiusSlrRad - data2.HostStarRadiusSlrRad.mean()) / data2.HostStarRadiusSlrRad.std()

data2.HostStarTempK = (data2.HostStarTempK - data2.HostStarTempK.mean()) / data2.HostStarTempK.std()

data2 = data2.drop(columns=data2.loc[:, data2.count() < 3584/2].columns)

data2['RadiusJpt'].fillna(np.mean(data2['RadiusJpt']), inplace=True)

data2['PeriodDays'].fillna(np.mean(data2['PeriodDays']), inplace=True)

data2['DistFromSunParsec'].fillna(np.mean(data2['DistFromSunParsec']), inplace=True)

data2['HostStarMassSlrMass'].fillna(np.mean(data2['HostStarMassSlrMass']), inplace=True)

data2['HostStarRadiusSlrRad'].fillna(np.mean(data2['HostStarRadiusSlrRad']), inplace=True)

data2['HostStarMetallicity'].fillna(np.mean(data2['HostStarMetallicity']), inplace=True)

data2['HostStarTempK'].fillna(np.mean(data2['HostStarTempK']), inplace=True)

data2 = data2.drop("RightAscension", 1)

data2 = data2.drop("Declination", 1)

pca = PCA()

pca.fit(data2)

X\_pca = pca.transform(data2)

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, data2.columns)))

X\_pca[:, :3].shape

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(3, c='b')

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

plt.xlabel('Количество признаков')

plt.ylabel('Описание дисперсии')

"""Вместо pyplot используем plotly """

import plotly.express as px

import plotly.graph\_objects as go

fig = px.scatter\_3d(X\_pca[:, :3], x=0, y=1, z=2, size\_max=18, opacity=0.7)

fig.show()

from sklearn.cluster import KMeans

inertia = []

for k in range(1, 20):

kmeans\_iter = KMeans(n\_clusters=k, random\_state=42).fit(data2)

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=10, random\_state=42)

# Проводим моделирование

kmeans.fit(data2)

all\_predictions = kmeans.predict(data2)

fig = px.scatter\_3d(X\_pca[:, :3], x=0, y=1, z=2, size\_max=18, opacity=0.7, color=all\_predictions)

fig.show()

fig = px.scatter\_3d(solar, x='0', y='1', z='2', size\_max=18, opacity=0.7, color = "class", text= "name")

fig.show()

result = X\_pca

solar\_indexes = [data2.index.get\_loc("Earth"), data2.index.get\_loc("Mercury"), data2.index.get\_loc("Venus"),

data2.index.get\_loc("Mars"), data2.index.get\_loc("Jupiter"), data2.index.get\_loc("Saturn"),

data2.index.get\_loc("Uranus"), data2.index.get\_loc("Neptune"), data2.index.get\_loc("Pluto")]

solar\_name = ["Earth", "Mercury", "Venus", "Mars", "Jupiter", "Saturn", "Uranus", "Neptune", "Pluto"]

earth = result[data2.index.get\_loc("Earth")]

mercury = result[data2.index.get\_loc("Mercury")]

venus = result[data2.index.get\_loc("Venus")]

mars = result[data2.index.get\_loc("Mars")]

jupiter = result[data2.index.get\_loc("Jupiter")]

saturn = result[data2.index.get\_loc("Saturn")]

uranus = result[data2.index.get\_loc("Uranus")]

neptune = result[data2.index.get\_loc("Neptune")]

pluto = result[data2.index.get\_loc("Pluto")]

solar = np.array([earth, mercury, venus, mars, jupiter, saturn, uranus, neptune, pluto])

solar = pd.DataFrame({'0': solar[:, 0], '1': solar[:, 1], '2':solar[:, 2]})

solar["index"] = np.NaN

solar["class"] = np.NaN

solar["name"] = np.NaN

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

solar["index"].iloc[i] = solar\_indexes[i]

solar['class'].iloc[i] = all\_predictions[i]

solar["name"].iloc[i] = solar\_name[i]

solar

clusters = {}

all\_predictions

for i in range(data2.shape[0]):

color = all\_predictions[i]

if clusters.get(color, False): clusters[color].append(data2.index[i])

else: clusters[color] = [data2.index[i]]

clusters.keys()

data2.loc[clusters[2], :]

ProfileReport(data2.loc[clusters[0], :])

ProfileReport(data2.loc[clusters[1], :])

ProfileReport(data2.loc[clusters[2], :])

ProfileReport(data2.loc[clusters[3], :])

ProfileReport(data2.loc[clusters[4], :])

ProfileReport(data2.loc[clusters[5], :])

ProfileReport(data2.loc[clusters[6], :])

ProfileReport(data2.loc[clusters[7], :])

ProfileReport(data2.loc[clusters[8], :])

ProfileReport(data2.loc[clusters[9], :])