## Задача 1

Условие

Набор данных: swiss.

Объясняемая переменная: *Agriculture*.

Регрессоры: *Education, Catholic.*

1. Оцените среднее значение, дисперсию и СКО переменных, указанных во втором и третьем столбце.
2. Постройте зависимости вида y = a + bx, где y – объясняемая переменная, x – регрессор.
3. Оцените, насколько «хороша» модель по коэффициенту детерминации R2?
4. Оцените, есть ли взаимосвязь между объясняемой переменной и объясняющей переменной (по значению p-статистики, «количеству звездочек» у регрессора в модели).

Решение

1. Оценим средние значения, дисперсии и СКО.   
   В результате имеем:

* Среднее значение *Agriculture* = 50.66
* Среднее значение *Catholic* = 41.14
* Среднее значение *Education* = 10.98
* Дисперсия *Agriculture* = 515.79
* Дисперсия *Catholic* = 1739.29
* Дисперсия *Education* = 92.45
* СКО *Agriculture* = 22.71
* СКО *Catholic* = 41.7
* СКО *Education* = 9.61

1. Построим линейные зависимости. В результате для первой и второй модели имеем:
   * A = -1.51\*ed + 67.24 для *Agriculture~Education*
   * A = 41.6728 + 0.2184\*cat для *Agriculture~Catholic*
2. Рассмотрим R2. Видим, что R2 у первой модели = 0.409 – это значит, что модель 1 объясняет 40% колебаний переменной *Agriculture* – меньше половины, нужны более сложные зависимости, но можно сделать вывод, что чем выше доля образованного населения, тем меньше людей занимаются обработкой земли. Для второй модели R2 = 0.16 – очень низкий показатель, делать вывод нецелесообразно.
3. P-статистика для первой модели имеем очень низкие показатели (3 звёздочки у каждого из параметров), что означает наличие четкой зависимости между параметрами и объясняемой переменной. У второй модели регрессор *Catholic* имеет только 2 звёздочки (большее значение p-статистики) – это значит, что зависимость между ним и *Agriculture* не такая сильная, но она всё ещё есть.

Выводы

Были построены две зависимости с объясняемой переменной Agriculture и разными объясняющими переменными: Education, Catholic. В первой модели есть причинно-следственная связь между поведением переменных, но она требует дополнительных регрессоров. Тоже самое, можно сказать и про вторую модель.

Приложение 1

Код решения задачи:

library("lmtest")

library("GGally")

library("car")

#Вариант 2 Бундуки Владислав Вячеславович; КМБО-06-20

data = swiss

help(swiss)

# \_1 - от Education

modele\_1 = lm(Agriculture~Education, data)

modele\_1

summary(modele\_1)

#Вывод:

# R^2 = 0.409 -> довольно низкий показатель, нужны более сложные зависимости

# A = -1.51\*ed + 67.24 -> чем выше доля образованного населения, тем меньше людей #занимаются обработкой земли

# Много звездочек => очень четкая зависимость проглядывается

plot(modele\_1) + abline(a = 67.24, b = -1.5, col = "red")

# \_2 - от catholic

modele\_2 = lm(Agriculture~Catholic, data)

modele\_2

summary(modele\_2)

#R^2 = 0.16 -> делать вывод - не целесообразно

# чем больше Католиков, тем больше людей занимаются обработкой земли

# Много звездочек => очень четкая зависимость проглядывается

plot(modele\_2) + abline(a = 41.67, b = 0.21, col = "red")

# 515.79 - большой разброс

var(data$Agriculture)

# 22.71 - СКО

sd(data$Agriculture)

# 50.66 - среднее значение

mean(data$Agriculture)

# 1739.29 - ОЧЕНЬ БОЛЬШОЙ РАЗБРОС

var(data$Catholic)

#41.7 - СКО

sd(data$Catholic)

# 41.14 - среднее значение

mean(data$Catholic)

# 92.45 - маленький разброс значений

var(data$Education)

# 9.61 - СКО

sd(data$Education)

# 10.98 - среднее значение

mean(data$Education)

## Задача 2.1

Условие

Набор данных: swiss.

Объясняемая переменная: *Agriculture*.

Регрессоры: *Fertility, Catholic, Education*.

1. Проверьте, что в наборе данных нет линейной зависимости (построить зависимости между переменными, указанными в варианте, и проверить, что R2 в каждой из них невысокий). В случае, если R2 большой, один из таких столбцов можно исключить из рассмотрения.
2. Постройте линейную модель зависимой переменной от указанных в варианте регрессоров по методу наименьших квадратов (команда lm пакета lmtest в языке R). Оценить, насколько хороша модель, согласно: 1) R2, 2) p-значениям каждого коэффициента.
3. Введите в модель логарифмы регрессоров (если возможно). Сравнить модели и выбрать наилучшую.
4. Введите в модель всевозможные произведения пар регрессоров, в том числе квадраты регрессоров. Найдите одну или несколько наилучших моделей по доле объяснённого разброса в данных R2.

Решение

1. Проверим линейную регрессию *Fertility ~ Catholic, Education. VIF < 6 =>* нет линейной зависимости между регрессорами и их можно использовать вместе.
2. Построим модель *Agriculture ~ Fertility, Education, Catholic*, чтобы посмотреть R2 и p-статистику. В результате её выполнения видим:
   * R2 = 0.582
   * У Fertility p-характеристика выше, чем у Education и Catholic.
3. Чтобы улучшить показатели модели, попробуем ввести в модель логарифмы, не забывая проверять отсутствие линейной зависимости. Подробный код поиска наилучшей модели приведён в Приложении 2.

Из всех моделей с логарифмами лучшей оказалась *modele\_log\_5 = lm(Agriculture ~ log(Fertility) + Education + Catholic, data)*, имеющая R2 = 0.5821 – больше, чем у исходной модели.

1. Попробуем тогда ввести в модель всевозможные произведения пар регрессоров, не забывая проверять отсутствие линейной зависимости. Подробный код поиска наилучшей модели приведён в Приложении 2.

Наилучшей среди моделей оказалась *model\_fin\_6 = lm(Agriculture ~ I(Fertility^2) + I(Education^2) + I(Catholic^2) + I(Fertility\*Education) + I(Fertility\*Education\*Catholic), data)*, имеющая R2 = 0.5971.

Выводы

Лучшей моделью оказалась модель, *Agriculture ~ I(Fertility^2), I(Education^2), I(Catholic^2), I(Fertility\*Education), I(Fertility\*Education\*Catholic*, так как все значения VIF регрессоров < 6.

Приложение 2

Код решения задачи:

library("lmtest")

library("GGally")

library("car")

data = swiss

help(swiss)

#Вариант 2 Бундуки Владислав КМБО-06-20

#Объясняемая переменная Agriculture , регрессоры: Fertility, Education, Catholic

model = lm(Agriculture ~ Fertility + Education + Catholic, data)

# 1) проверка на линейную зависимость среди регрессоров

model\_t = lm(Fertility ~ Catholic + Education, data)

vif(model\_t)

#Education Catholic

#1.024247 1.024247

#проверим R^2

model\_1 = lm(Agriculture ~ Fertility + Education, data)

summary(model\_1)

#Arg = 85.7253 + (-0.2)\*Fer + (-1.7)\*Edu

#R^2 = 0.41 , p-характеристика = 6.707e-06

#R^2 низкий => нет строгой линейной зависимости

model\_2 = lm(Agriculture ~ Fertility + Catholic, data)

summary(model\_2)

#Arg = 16.7384 + (0.38)\*Fer + (0.16)\*Cat

#R^2 = 0.2 , p-характеристика = 0.008134

#R^2 очень низкий => переменные почти не связаны между собой

#p-характеристика высокая

model\_3 = lm(Agriculture ~ Education + Catholic, data)

summary(model\_3)

#Arg = 59.06 + (-1.4)\*Edu + (0.17)\*Cat

#R^2 = 0.5 , p-характеристика = 2.104e-07

#R^2 низкий => нет строгой линейной зависимости

# 2)Строим линейную модель по 3-ем регрессорам

model

summary(model)

#Agriculture = 117.2729 + (-0.78)\*Fertility + (-2.01)\*Edu + (0.26)\*Cat

#R^2 = 0.582 , p = 2.956e-08

#R^2 не высокий => нельзя сделать корректные выводы

# p значение низкое => реальные данные мало будут отличаться от данных модели

# р-характеристика у Fertility = (\*\*)

# р-характеристика у Education = (\*\*\*)

# р-характеристика у Catholic = (\*\*\*)

# 3)Введем в модель логарифмы регрессоров и сравним их для поиска лучшей

modele\_log\_1 = lm(Agriculture ~ log(Fertility) + log(Education) + log(Catholic), data)

summary(modele\_log\_1)

vif(modele\_log\_1)

#R^2 = 0.50 - модель ухудшилась

#Agriculture = 120.271 + (-8.717)\*Fertility + (-20.352)\*Edu + ( 3.426 )\*Cat

#log(Fertility) log(Education) log(Catholic)

#1.605793 1.490099 1.103041

modele\_log\_2 = lm(Agriculture ~ log(Fertility) + log(Education) + Catholic, data)

summary(modele\_log\_2)

vif(modele\_log\_2)

#R^2 = 0.5394 - модель ухудшилась

#Agriculture = 145.97625 + (-14.4779)\*Fertility + (-19.54853)\*Edu + (0.17028 )\*Cat

#log(Fertility) log(Education) Catholic

#1.656365 1.458331 1.184675

modele\_log\_3 = lm(Agriculture ~ log(Fertility) + Education + log(Catholic), data)

summary(modele\_log\_3)

vif(modele\_log\_3)

#R^2 = 0.5246 - модель ухудшилась

#Agriculture = 255.9569 + (-46.5763)\*Fertility + (-2.2245 )\*Edu + (5.5737 )Cat

#log(Fertility) Education log(Catholic)

#2.616193 2.426323 1.216809

modele\_log\_4 = lm(Agriculture ~ Fertility + log(Education) + log(Catholic), data)

summary(modele\_log\_4)

vif(modele\_log\_4)

#R^2 = 0.5018 - модель ухудшилась

#Agriculture = 97.0689 + (-0.1948)\*Fertility + (-20.7370)\*Edu + (3.6925)Cat

# Fertility log(Education) log(Catholic)

#1.626027 1.429427 1.180360

modele\_log\_5 = lm(Agriculture ~ log(Fertility) + Education + Catholic, data)

summary(modele\_log\_5)

vif(modele\_log\_5)

#R^2 = 0.5821 - модель улудшилась

#Agriculture = 286.99567 + (-52.62785)\*Fertility + (-2.13538)\*Edu + ( 0.24021)\*Cat

#log(Fertility) Education Catholic

#2.606621 2.255065 1.241080

modele\_log\_6 = lm(Agriculture ~ Fertility + Education + log(Catholic), data)

summary(modele\_log\_6)

vif(modele\_log\_6)

#R^2 = 0.5199 - модель ухудшилась

#Agriculture = 103.3117 + (-0.6653)\*Fertility + ( -2.1044)\*Edu + (5.8601)Cat

#Fertility Education log(Catholic)

#2.345216 2.060489 1.312589

modele\_log\_7 = lm(Agriculture ~ Fertility + log(Education) + Catholic, data)

summary(modele\_log\_7)

vif(modele\_log\_7)

#R^2 =0.5469 - модель ухудшилась

#Agriculture = 106.48306 + (-0.30771)\*Fertility + (-19.93094)\*Edu + (0.18471)Cat

# Fertility log(Education) Catholic

#1.690517 1.383282 1.277755

#Из всех моделей только у modele\_log\_5 улучшился R^2 на 0.001

# 4) Введем в модель всевозможные произведения пар регрессоров, в том числе и квадраты регрессоров

model\_fin\_1 = lm(Agriculture ~ Fertility + Education + Catholic + I(Fertility^2) + I(Education^2) + I(Catholic^2) + I(Fertility\*Education) + I(Fertility\*Catholic) + I(Education\*Catholic) + I(Fertility\*Education\*Catholic), data)

summary(model\_fin\_1) #R^2 = 0.649

vif(model\_fin\_1)

model\_fin\_2 = lm(Agriculture ~ Education + Catholic + I(Fertility^2) + I(Education^2) + I(Catholic^2) + I(Fertility\*Education) + I(Fertility\*Catholic) + I(Education\*Catholic) + I(Fertility\*Education\*Catholic), data)

summary(model\_fin\_2)#R^2 = 0.6438

vif(model\_fin\_2)

model\_fin\_3 = lm(Agriculture ~ Education + Catholic + I(Fertility^2) + I(Education^2) + I(Catholic^2) + I(Fertility\*Education) + I(Education\*Catholic) + I(Fertility\*Education\*Catholic), data)

summary(model\_fin\_3)#R^2 = 0.6347

vif(model\_fin\_3)

model\_fin\_4 = lm(Agriculture ~ Catholic + I(Fertility^2) + I(Education^2) + I(Catholic^2) + I(Fertility\*Education) + I(Education\*Catholic) + I(Fertility\*Education\*Catholic), data)

summary(model\_fin\_4)#R^2 = 0.6239

vif(model\_fin\_4)

model\_fin\_5 = lm(Agriculture ~ I(Fertility^2) + I(Education^2) + I(Catholic^2) + I(Fertility\*Education) + I(Education\*Catholic) + I(Fertility\*Education\*Catholic), data)

summary(model\_fin\_5)#R^2 = 0.6

vif(model\_fin\_5)

model\_fin\_6 = lm(Agriculture ~ I(Fertility^2) + I(Education^2) + I(Catholic^2) + I(Fertility\*Education) + I(Fertility\*Education\*Catholic), data)

#Agriculture = 7.919e+01 + (-3.195e-03)\*Fertility^2 + (-8.119e-03)\*Edu^2 + (2.724e-03)Cat^2 #+ (-2.640e-02)\*(Fertility\*Education) + (-6.148e-05)\*(Fertility\*Education\*Catholic)

summary(model\_fin\_6)#R^2 = 0.5971

vif(model\_fin\_6)

#Вывод: Agriculture = 7.919e+01 + (-3.195e-03)\*Fertility^2 + (-8.119e-03)\*Edu^2 + (2.724e-#03)Cat^2 + (-2.640e-02)\*(Fertility\*Education) + (-6.148e-05)\*(Fertility\*Education\*Catholic)

#это лучшая модель ,т.к. VIF < 6

## Задача 2.2

Условие

Набор данных: swiss.

Объясняемая переменная: *Agriculture*.

Регрессоры: *Fertility, Catholic, Education*.

Для зависимости, построенной при решении практического задания №2, оцените:

1. Доверительные интервалы для всех коэффициентов в модели, p = 95%.
2. Сделайте вывод о отвержении или невозможности отвергнуть статистическую гипотезу о том, что коэффициент равен 0
3. Доверительный интервал для одного прогноза (p = 95%, набор значений регрессоров выбираете сами).

Решение

Имеем следующую модель:

Характеристики модели зависимости параметра: *Agriculture* от параметров *Fertility*, *Education*, *Catholic* в наборе данных Swiss. Зная стандартные ошибки (1.058e+01 для свободного коэффициента, 1.952e-03 для *Fertility^2*, 1.090e-03 для *Catholic^2*, 8.821e-03 для *Education^2,* 9.636e-03 для *Fertility\*Education*, 1.638e-04 для *Fertility \* Education \* Catholic*) найдём доверительные интервалы для коэффициентов по формуле Estimate t-критерий Стьюдента \* Std. Error: (*см. Таблица 1*)

* *Fertility^2*:

Критерий Стьюдента(t) = 2.03

Доверительный интервал: [-0.007157706, 0.0007678357]

* *Edecation^2*:

Критерий Стьюдента(t) = 2.03

Доверительный интервал: [-0.0260264, 0.009788768]

* *Catholic^2*:

Критерий Стьюдента(t) = 2.03

Доверительный интервал: [0.0005116675, 0.004937303]

* *Fertility\*Education*:

Критерий Стьюдента(t) = 2.03

Доверительный интервал: [-0.04595943, -0.006835191]

* *Fertility\*Education\*Catholic*:

Критерий Стьюдента(t) = 2.03

Доверительный интервал: [-0.0003940142, 0.0002710492]

* *Intercept*:

Критерий Стьюдента(t) = 2.03

Доверительный интервал: [57.71211, 100.6692]

1. Оценим доверительные интервалы для всех коэффициентов в модели (для p=95%):

Число степеней свободы в модели df = 41 - 6 = 35, и t-критерий Стьюдента тогда равен 2.03.

Таблица 1. Характеристики модели зависимости параметра Agriculture от параметров Fertility, Education, Catholic в наборе данных Swiss

|  |  |  |  |
| --- | --- | --- | --- |
| Параметр \ Характеристики | Значение | Доверительный интервал | “Может ли коэффициент быть равен 0?” |
| (Intercept) | 79.19065 | [57.7, 100.7] | Нет |
| *Fertility^2* | -0.003194935 | [-0.007, 0.0008] | Да |
| *Education^2* | -0.008118814 | [-0.03, 0.01] | Да |
| *Catholic^2* | 0.002724485 | [0.0005, 0.005] | Нет |
| *Fertility\*Education* | -0.02639731 | [-0.046, -0.007] | Нет |
| *Fertility\*Education\*Catholic* | -6.14825e-05 | [-0.0004, 0.0003] | Да |

1. По доверительным интервалам сделаем вывод об отвержении или невозможности отвергнуть статистическую гипотезу о том, что коэффициент равен 0:

* Доверительный интервал свободного коэффициента: [57.7, 100.7], 0 не попадает в доверительный интервал, поэтому отвергаем статистическую гипотезу о том, что коэффициент равен 0.
* Доверительный интервал *Fertility*: [-0.007, 0.0008], 0 попадает в доверительный интервал, поэтому не отвергаем статистическую гипотезу о том, что коэффициент равен 0.
* Доверительный интервал *Catholic*: [0.0005, 0.005], 0 не попадает в доверительный интервал, поэтому отвергаем статистическую гипотезу о том, что коэффициент равен 0.
* Доверительный интервал *Education*: [-0.03, 0.01], 0 попадает в доверительный интервал, поэтому не отвергаем статистическую гипотезу о том, что коэффициент равен 0.
* Доверительный интервал *Education\*Fertility*: [-0.046, -0.007], 0 не попадает в доверительный интервал, поэтому отвергаем статистическую гипотезу о том, что коэффициент равен 0.
* Доверительный интервал *Education\*Fertility\*Catholic*: [-0.0004, 0.0003], 0 попадает в доверительный интервал, поэтому не отвергаем статистическую гипотезу о том, что коэффициент равен 0.

1. Оценим доверительный интервал для одного прогноза (p = 95%, *Fertility = 15, Education = 15, Catholic = 30*):

  
Рисунок 1. Оценка доверительного интервалa.

Имеем доверительный интервал [53.96525,91.52014]

Полный код решения задачи приведён в Приложении 3.

Выводы*:*

Поскольку 0 попадает в доверительный интервал регрессоров*,* то объясняющая переменная (Agriculture) не связана с объясняемыми переменными: *Fertility, Education.* Доверительный интервал для ненулевых регрессоров имеет маленький разброс, следовательно - они имеют сильную взаимосвязь с объясняемой переменной.

Прогноз модели Agriculture ~ Fertility, Education, Catholic оценивается как 72.74269 Доверительный интервал для свободного коэффициента имеет вид: [54, 91.5].

Приложение 3

Код решения задачи:

library("lmtest")

library("GGally")

library("car")

data = swiss

help(swiss)

#Вариант 2 Бундуки Владислав КМБО-06-20

#Объясняемая переменная Agriculture , регрессоры: Fertility, Education, Catholic

#Используем лучшую модель, основываясь на результате работы 2.1 :

#"Agriculture ~ I(Fertility^2), I(Education^2), I(Catholic^2), I(Fertility\*Education), I(Fertility\*Education\*Catholic)"

model = lm(Agriculture ~ I(Fertility^2)+ I(Education^2)+ I(Catholic^2)+ I(Fertility\*Education)+ I(Fertility\*Education\*Catholic), data)

model

summary(model)

#Agriculture = 7.919e+01 + (-3.195e-03)\*Fertility^2 + (-8.119e-03)\*Edu^2 + (2.724e-03)Cat^2 + (-2.640e-02)Fertility\*Education +(-6.148e-05)Fertility \* Education \* Catholic

#R^2 = 0.5971

# 43 наблюдений и оценивалось 4 коэффициента: 41 - 6 = 35 степени свободы;

#Оценим доверительные интервалы для Agriculture:

# a) Доверительный интервал для Fertility^2

Std\_Err = 1.952e-03

# Критерий Стьюдента: 95%, 35 степени свободы

t\_critical = qt(0.975, df = 35) # ~2.03

model$coefficients[2] - t\_critical \* Std\_Err

model$coefficients[2] + t\_critical \* Std\_Err

# Доверительный интервал для Fertility = [model$coefficients[2] - t\_critical \* Std\_Err , model$coefficients[2] + t\_critical \* Std\_Err]

# [-0.007157706 , 0.0007678357] -> коэффициент может быть равным 0

# Проверка:

confint(model, level = 0.95)

# 2.5 % 97.5 %

#(Intercept) 57.8162199031 1.005651e+02

#I(Fertility^2) -0.0071360925 7.462226e-04

#I(Education^2) -0.0259326469 9.695019e-03

#I(Catholic^2) 0.0005238842 4.925086e-03

#I(Fertility \* Education) -0.0458579729 -6.936649e-03

#I(Fertility \* Education \* Catholic) -0.0003922415 2.692765e-04

# б) Доверительный интервал для Education^2

Std\_Err = 8.821e-03

# Критерий Стьюдента: 95%, 35 степени свободы

model$coefficients[3] - t\_critical \* Std\_Err

model$coefficients[3] + t\_critical \* Std\_Err

# Доверительный интервал для Education = [model$coefficients[3] - t\_critical \* Std\_Err , model$coefficients[3] + t\_critical \* Std\_Err]

# [-0.0260264 , 0.009788768] -> коэффициент может быть равным 0

# в) Доверительный интервал для Catholic^2

Std\_Err = 1.090e-03

# Критерий Стьюдента: 95%, 35 степени свободы

model$coefficients[4] - t\_critical \* Std\_Err

model$coefficients[4] + t\_critical \* Std\_Err

# Доверительный интервал для Catholic = [model$coefficients[4] - t\_critical \* Std\_Err , model$coefficients[4] + t\_critical \* Std\_Err]

# [0.0005116675 , 0.004937303] -> коэффициент не может быть равным 0

# г) доверительный интервал для Fertility \* Education:

Std\_Err = 9.636e-03

# Критерий Стьюдента: 95%, 35 степени свободы

model$coefficients[5] - t\_critical \* Std\_Err

model$coefficients[5] + t\_critical \* Std\_Err

# Доверительный интервал для Fertility \* Education = [model$coefficients[5] - t\_critical \* Std\_Err , model$coefficients[5] + t\_critical \* Std\_Err]

# [-0.04595943 , -0.006835191] -> коэффициент не может быть равным 0

# д) доверительный интервал для Fertility \* Education \* Catholic:

Std\_Err = 1.638e-04

# Критерий Стьюдента: 95%, 35 степени свободы

model$coefficients[6] - t\_critical \* Std\_Err

model$coefficients[6] + t\_critical \* Std\_Err

# Доверительный интервал для Fertility \* Education \* Catholic = [model$coefficients[6] - t\_critical \* Std\_Err , model$coefficients[6] + t\_critical \* Std\_Err]

# [-0.0003940142 , 0.0002710492] -> коэффициент может быть равным 0

# е) доверительный интервал для свободного коэф:

Std\_Err = 1.058e+01

# Критерий Стьюдента: 95%, 35 степени свободы

model$coefficients[1] - t\_critical \* Std\_Err

model$coefficients[1] + t\_critical \* Std\_Err

# Доверительный интервал для Intercept = [model$coefficients[1] - t\_critical \* Std\_Err , model$coefficients[1] + t\_critical \* Std\_Err]

# [57.71211 , 100.6692] -> коэффициент не может быть равным 0

# Вывод: Поскольку 0 попадает в доверительный интервал регрессоров => не все регрессоры связаны с объясняемой переменной.

#Построим доверительный интервал для прогноза (Объясняемая переменная: Agriculture, регрессоры: Fertility , Education , Catholic )

model = lm(Agriculture ~ I(Fertility^2)+ I(Education^2)+ I(Catholic^2)+ I(Fertility\*Education)+ I(Fertility\*Education\*Catholic), data)

model

summary(model)

#Agriculture = 7.919e+01 + (-3.195e-03)\*Fertility^2 + (-8.119e-03)\*Edu^2 + (2.724e-03)Cat^2 + (-2.640e-02)Fertility\*Education +(-6.148e-05)Fertility \* Education \* Catholic

new.data = data.frame(Fertility = 15, Education = 15, Catholic = 30)

predict(model, new.data, interval = "confidence")

# fit lwr upr

# 72.74269 53.96525 91.52014

# Вывод: Прогноз модели оценивается как 72.74269;

# Доверительный интервал для свободного коэффициентa = [53.96525,91.52014]

## Задача 3

Условие

Набор данных: r22i\_os26c.sav.

Объясняемая переменная: заработная плата – *salary.*

Регрессоры: возраст, пол, образование, населенный пункт, продолжительность рабочей недели, семейное положение (состоит ли в зарегистрированном браке/никогда не был в браке / разведён или вдовец /живет вместе, но не зарегистрированы /зарегистрированы, но не живут вместе), населенный пункт (областной центр/город), продолжительность рабочей недели, – *salary*, *age, sex, r\_educ, status1, dur, wed, wed2, wed3.*

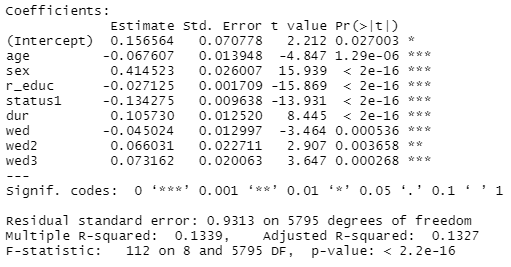
1. Постройте линейную регрессию зарплаты на все параметры, которые Вы выделили   
   из данных мониторинга. Не забудьте оценить коэффициент вздутия дисперсии VIF.
2. Поэкспериментируйте с функциями вещественных параметров: используйте  
   логарифм и степени (хотя бы от 0.1 до 2 с шагом 0.1).
3. Выделите наилучшие модели из построенных: по значимости параметров,  
   включённых в зависимости, и по объяснённому с помощью построенных  
   зависимостей разбросу adjusted R2 – R2adj.
4. Сделайте вывод о том, какие индивиды получают наибольшую зарплату.
5. Оцените регрессии для подмножества индивидов: а) городские жители, не состоявшие в браке; б) разведенные женщины, без высшего образования

Решение

Считаем из файла данные исследования и представим их в удобном виде:

1. *salary* – переменная с нормализованной зарплатой (формула для нормализации значения: ((salary - mean(salary)) / sqrt(var(salary)) )
2. *age* ­– переменная с нормализованным возрастом.
3. Переменная *sex* принимает 1, если человеком был мужчина и 0, если человеком была женщина.
4. *r\_edu*c = 1, если у человека есть высшее образование.
5. *status1* = 1, если человек живёт в городе или в областном центре.
6. *dur* – продолжительность рабочей недели.
7. Семейное положение:
   * *wed* = 1, если человек никогда не был в браке или состоял в зарегистрированном браке.
   * *wed2* = 1, если человек разведён или вдовец
   * *wed3* = 1, если люди жили вместе, но не были зарегистрированы или официально были зарегистрированы, но вместе не проживали.
   * Проверим, что между *wed, wed2, wed3* нет линейной зависимости.
8. Построим линейную регрессию зарплаты на все параметры:

Строим модель salary~age + sex + r\_educ + status1 + dur + wed + wed2 + wed3

  
Рисунок 2. Характеристики model, где model = lm(data = data3, salary~age + sex + r\_educ + status1 + dur + wed + wed2 + wed3)

Оценим vif у модели:

  
Рисунок 3. Результат проверки нашей модели на линейную зависимость среди регрессоров.

Из рисунка 2 видим, что vif низкий – линейной зависимости между регрессорами нет.

1. Введём в модель степени и логарифмы.

Модели со степенями, где степень изменяется от 0.1 до 2, с шагом в 0.1:

Степень 0.1:

*model\_test\_1 = lm(data = data3, salary~ age + sex + r\_educ + status1 + dur + wed + wed2 + wed3 + I(dur^0.1) + I(age^0.1))* - у это модели vif меньше 7, низкие показатели p-статистики и R^2 = 0.1691.

Степень 0.2:

model\_test\_1 = lm(data = data3, salary~ age + sex + r\_educ + status1 + dur + wed + wed2 + wed3 + I(dur^0.2) + I(age^0.2)) - у это модели vif меньше 7, низкие показатели p-статистики и R^2 = 0.1695.

Степень 0.3:

model\_test\_1 = lm(data = data3, salary~ age + sex + r\_educ + status1 + dur + wed + wed2 + wed3 + I(dur^0.3) + I(age^0.3)) - у это модели vif ~10, значит есть линейная зависимость.

Степень 0.4:

model\_test\_1 = lm(data = data3, salary~ age + sex + r\_educ + status1 + dur + wed + wed2 + wed3 + I(age^0.4) + I(dur^0.4)) - у это модели vif ~13, значит есть линейная зависимость.

Так как при степень большей 0.3 появляется линейная зависимость, проверим модель сразу на высокой степени (2):

model\_test\_1 = lm(data = data3, salary~ age + sex + r\_educ + status1 + dur + wed + wed2 + wed3 + I(age^2) + I(dur^2)) - у это модели vif меньше 7, неплохие показатели р-статистики и R^2 = 0.1488.

Модель с логарифмами:

model\_test\_1 = lm(data = data3, salary~ age + sex + r\_educ + status1 + dur + wed + wed2 + wed3 + I(log(age)) + I(log(dur))) - у это модели vif меньше 7 и R^2 = 0.1687.

Попробуем ввести логарифм только для одного из двух регрессоров:

1. model\_test\_1 = lm(data = data3, salary~ age + sex + r\_educ + status1 + dur + wed + wed2 + wed3 + I(log(dur))) - у это модели vif меньше 7, р-статистика хорошая, кроме dur и log(dur), но если мы уберем log(dur), то получим исходную модель и R^2 = 0.1407.
2. model\_test\_1 = lm(data = data3, salary~ age + sex + r\_educ + status1 + dur + wed + wed2 + wed3 + I(log(age))) - у это модели vif меньше 7, р-статистика хорошая, кроме wed, wed2, wed3, но мы не можем их убрать, так как они важны для нашей модели; R^2 = 0.1596.

Из них лучший R2 имеет модель с log(dur), но у неё плохая p-статистика для dur и log(dur).

1. Выделим наилучшие модели из построенных: по значимости параметров, включённых в зависимости, и по объяснённому с помощью построенных зависимостей разбросу adjusted R2 – R2\_adj

Из всех моделей, без линейной зависимости, лучшие по значению R^2 – это модели: со степенью 0.4 и логарифмом(log(dur), lof(age)). Разброс R2 - R2\_adj у них одинаковый. Из этих моделей лучшей является модель model\_test\_1 = lm(data = data3, salary~ age + sex + r\_educ + status1 + dur + wed + wed2 + wed3 + I(log(age)) + I(log(dur))). Лучшей же моделью по р-статистике является наша исходная модель: model = lm(data = data3, salary~age + sex + r\_educ + status1 + dur + wed + wed2 + wed3).

1. Согласно наилучшей модели по р-статистике больше всего зарабатывают мужчины молодого возраста, без высшего образования, работающие в областном центре, перерабатывающие, не состоявшие в браке или вдовец или официально зарегистрированные, но не проживающие вместе.
2. Оценим регрессии для подмножества индивидов: а) Женщины не замужем б) Женщины, живущие в городе, разведенные

а) выделим множество женщин не замужем:

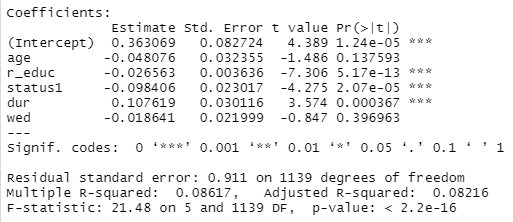
data4 = subset(data3, sex == 0)

data4

data5 = subset(data4, wed2 == 1)

data5

Имеем следующую модель:

  
Рисунок 4. Характеристики model\_subset, где model\_subset = lm(data = data5, salary~age + r\_educ + status1 + dur + wed)

R2 ~ 0.08617. Согласно этой модели наивысшую зарплату получают женщины молодого возраста, без высшего образования, работающие не в городе, перерабатывающие и не состоявшие в браке.

б) выделим множество разведенных женщин без высшего образования:

data6 = subset(data3, sex == 0)

data6

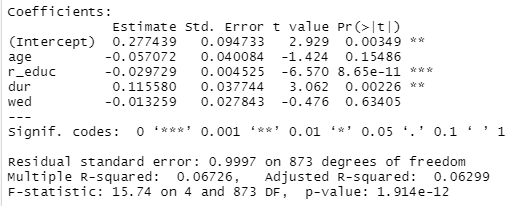
data7 = subset(data6, status1 == 1)

data7

data8 = subset(data7, wed2 == 1)

data8

Тогда имеем следующую модель:

   
Рисунок 5. Характеристики model\_subset, где model\_subset = lm(data = data8, salary~ age + r\_educ + dur + wed)

R2 ~ 0.06771. Согласно этой модели наивысшую зарплату получают женщины молодого возраста, живущие в городе, разведенные или которые никогда не были в браке, без высшего образования и перерабатывающие.

Полный код решения задачи приведён в Приложении 4.

Выводы

Из всех полученных данных наибольшую заработную плату имеют мужчины молодого возраста, без высшего образования, работающие в областном центре, перерабатывающие, не состоявшие в браке или вдовец, или официально зарегистрированные, но не проживающие вместе.

В подмножестве: «женщины не замужем», наивысшую зарплату получают женщины молодого возраста, без высшего образования, работающие не в городе, перерабатывающие и не состоявшие в браке.

В подмножестве: «женщины, живущие в городе, разведенные» наивысшую зарплату получают женщины молодого возраста, живущие в городе, разведенные или которые никогда не были в браке, без высшего образования и перерабатывающие.

Приложение 4

Код решения:

install.packages("devtools")

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

library("lmtest")

library("rlms")

library("dplyr")

library("GGally")

library("car")

library("sandwich")

data <- rlms\_read("C:\\Users\\vladb\\Downloads\\r22i\_os26c.sav")

glimpse(data)

data2 = select(data, rj13.2, r\_age, rh5, r\_educ, status, rj6.2, r\_marst)

#Убираем строки с NA

data2 = na.omit(data2)

glimpse(data2)

#зарплата c элементами нормализации

data2$rj13.2

sal = as.numeric(data2$rj13.2)

sal1 = as.character(data2$rj13.2)

sal2 = lapply(sal1, as.integer)

sal = as.numeric(unlist(sal2))

mean(sal)

data2["salary"] = (sal - mean(sal)) / sqrt(var(sal))

data2["salary"]

#возраст c элементами нормализации

age1 = as.character(data2$r\_age)

age2 = lapply(age1, as.integer)

age3 = as.numeric(unlist(age2))

data2["age"]= (age3 - mean(age3)) / sqrt(var(age3))

data2["age"]

#пол

#data2["sex"]=data2$rh5

#data2["sex"] = lapply(data2$rh5, as.character)

data2$sex = as.numeric(data2$rh5)

data2$sex[which(data2$sex!='1')] <- 0 #женский

data2$sex[which(data2$sex=='1')] <- 1 #мужской

#образование

#data2["r\_educ"] = data2$r\_educ

#data2["r\_educ"] = lapply(data2$r\_educ, as.character)

#data2["higher\_educ"] = data2$r\_educ

data2$r\_educ = as.numeric(data2$r\_educ)

data2$r\_educ[which(data2$r\_educ=='21')] <- 1 #есть диплом о высшем образовании

data2$r\_educ[which(data2$r\_educ=='22')] <- 1 #аспирантура и т.п. без диплома

data2$r\_educ[which(data2$r\_educ=='23')] <- 1 #аспирантура и т.п. с дипломом

#населенный пункт

#data2["status1"]=data2$status

#data2["status1"] = lapply(data2$status, as.character)

data2$status1 = as.numeric(data2$status)

data2$status1[which(data2$status1=='1')] <- 1 #областной центр

data2$status1[which(data2$status1=='2')] <- 1 #город

#продолжительность рабочей недели

dur1 = as.character(data2$rj6.2)

dur2 = lapply(dur1, as.integer)

dur3 = as.numeric(unlist(dur2))

data2["dur"] = (dur3 - mean(dur3)) / sqrt(var(dur3))

#семейное положение

#data2["wed"]= data2$r\_marst

#data2["wed"] = lapply(data2$r\_marst, as.character)

data2$wed = as.numeric(data2$r\_marst)

data2$wed[which(data2$wed=='1')] <- 1 #никогда не были в браке

data2$wed[which(data2$wed=='2')] <- 1 # Состоят в зарегистрированном браке

#data2["wed2"] = lapply(data2["wed"], as.character)

data2$wed2 = as.numeric(data2$r\_marst)

data2$wed2[which(data2$wed=='4')] <- 1 # Разведены

data2$wed2[which(data2$wed=='5')] <- 1 # Bдовец/вдова

#data2$wed2 = as.numeric(data2$wed2)

#data2["wed3"] = lapply(data2["wed"], as.character)

data2$wed3 = as.numeric(data2$r\_marst)

data2$wed3[which(data2$wed=='3')] <- 1 # Живете вместе, но не зарегистрированы

data2$wed3[which(data2$wed=='6')] <- 1 # ОФИЦИАЛЬНО ЗАРЕГИСТРИРОВАНЫ, НО ВМЕСТЕ НЕ ПРОЖИВАЮТ

data3 = select(data2, salary, age, sex, r\_educ, status1, dur, wed, wed2, wed3)

#построение зависимостей для данных

model = lm(data = data3, salary~age + sex + r\_educ + status1 + dur + wed + wed2 + wed3)

summary(model)

vif(model)

#R^2 = 0.1339 - очень низкий

#p-характеристика у всех значений (\*\*\*), кроме wed2 (\*\*)

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

model\_test\_1 = lm(data = data3, salary~ age + sex + r\_educ + status1 + dur + wed + wed2 + wed3 + I(dur^0.1) + I(age^0.1))

summary(model\_test\_1)

vif(model\_test\_1)

#R^2 = 0.1691

#р - ухудшился

#vif <= 7

model\_test\_1 = lm(data = data3, salary~ age + sex + r\_educ + status1 + dur + wed + wed2 + wed3 + I(dur^0.2) + I(age^0.2))

summary(model\_test\_1)

vif(model\_test\_1)

#R^2 = 0.1695

#р - ухудшился

#vif <= 7

model\_test\_1 = lm(data = data3, salary~ age + sex + r\_educ + status1 + dur + wed + wed2 + wed3 + I(dur^0.3) + I(age^0.3))

summary(model\_test\_1)

vif(model\_test\_1)

#R^2 = 0.17

#р - ухудшился

#vif <= 10 ~ есть линейнай зависимость

model\_test\_1 = lm(data = data3, salary~ age + sex + r\_educ + status1 + dur + wed + wed2 + wed3 + I(age^0.4) + I(dur^0.4))

summary(model\_test\_1)

vif(model\_test\_1)

#R^2 = 0.1704

#р - ухудшился

#vif <= 13 ~ есть линейнай зависимость

model\_test\_1 = lm(data = data3, salary~ age + sex + r\_educ + status1 + dur + wed + wed2 + wed3 + I(dur^0.4))

summary(model\_test\_1)

vif(model\_test\_1)

#R^2 = 0.1413

#р - ухудшился

#vif <= 14 ~ есть линейнай зависимость

model\_test\_1 = lm(data = data3, salary~ age + sex + r\_educ + status1 + dur + wed + wed2 + wed3 + I(age^0.4))

summary(model\_test\_1)

vif(model\_test\_1)

#R^2 = 0.1595

#р - ухудшился

#vif <= 11 ~ есть линейнай зависимость

model\_test\_1 = lm(data = data3, salary~ age + sex + r\_educ + status1 + dur + wed + wed2 + wed3 + I(dur^0.4))

summary(model\_test\_1)

vif(model\_test\_1)

#R^2 = 0.1413

#р - ухудшился

#vif <= 14 ~ есть линейнай зависимость

model\_test\_1 = lm(data = data3, salary~ age + sex + r\_educ + status1 + dur + wed + wed2 + wed3 + I(age^2) + I(dur^2))

summary(model\_test\_1)

vif(model\_test\_1)

#R^2 = 0.1488

#р - ухудшился

#vif <= 3

model\_test\_1 = lm(data = data3, salary~ age + sex + r\_educ + status1 + dur + wed + wed2 + wed3 + I(dur^2))

summary(model\_test\_1)

vif(model\_test\_1)

#R^2 = 0.1345

#р - ухудшился

#vif <= 3

model\_test\_1 = lm(data = data3, salary~ age + sex + r\_educ + status1 + dur + wed + wed2 + wed3 + I(age^2))

summary(model\_test\_1)

vif(model\_test\_1)

#R^2 = 0.1484

#р - чуть-чуть хуже, чем у исходной модели

#vif <= 3

model\_test\_1 = lm(data = data3, salary~ age + sex + r\_educ + status1 + dur + wed + wed2 + wed3 + I(log(age)) + I(log(dur)))

summary(model\_test\_1)

vif(model\_test\_1)

#R^2 = 0.1687

#р - ухудшился

#vif <= 7

model\_test\_1 = lm(data = data3, salary~ age + sex + r\_educ + status1 + dur + wed + wed2 + wed3 + I(log(dur)))

summary(model\_test\_1)

vif(model\_test\_1)

#R^2 = 0.1407

#р - хуже, чем у начальной модели, но лучше, чем у остальных моделей со степенями и логарифмами

#vif <= 5

model\_test\_1 = lm(data = data3, salary~ age + sex + r\_educ + status1 + dur + wed + wed2 + wed3 + I(log(age)))

summary(model\_test\_1)

vif(model\_test\_1)

#R^2 = 0.1596

#р - ухудшился

#vif <= 7

#Итог: лучшая модель это salary~ age + sex + r\_educ + status1 + dur + wed + wed2 + wed3 + I(log(dur))

#проверим модель на линейную зависимость параметров

modele\_1 = lm(dur~I(log(dur)), data3)

modele\_1

summary(modele\_1) # R^2 = 0.8063 < 0.8, значит нет линейной зависимости и можно использовать в одной модели

modele\_2 = lm(age~I(dur^2), data3)

modele\_2

summary(modele\_2) # R^2 = 3.001e-05 < 0.1 , значит нет линейной зависимости и можно использовать в одной модели

#Вывод: более высокую ЗП получают мужчины молодого возраста без высшего образования, работающие в областном центре , перерабатывающие,

# не состоявшие в браке или вдовец или официально зарегистрированные, но не проживающие вместе.

#(Intercept) age sex r\_educ status1 dur wed wed2 wed3 I(log(dur))

#0.009378 -0.100804 0.452144 -0.030679 -0.154806 0.068537 -0.086875 0.143205 0.152724 0.056648

#Ищем подмножества: "Женщины не замужем"

data4 = subset(data3, sex == 0)

data4

data5 = subset(data4, wed2 == 1)

data5

# Ищем подмножество: "женщины, живущие в городе, разведенные"

data6 = subset(data3, sex == 0)

data6

data7 = subset(data6, status1 == 1)

data7

data8 = subset(data7, wed2 == 1)

data8

#так как регрессор sex = const и wed2 = const, то уберем их из нашей модели, потому что строить зависимость от const не имеет смысла; так как wed = wed3 , то уберем wed3 из нашей модели.

model\_subset = lm(data = data5, salary~age + r\_educ + status1 + dur + wed)

summary(model\_subset)

# R^2 = 0.08617

# Наивысшую зарплату получают женщины молодого возраста, без высшего образования, работающие не в городе, перерабатывающие и не состоявшие в браке

#так как регрессор sex = const и wed2 = const, status1 = const, то уберем их из нашей модели, потому что строить зависимость от const не имеет смысла; так как wed = wed3 , то уберем wed3 из нашей модели.

model\_subset = lm(data = data8,salary~age + r\_educ + dur + wed)

summary(model\_subset)

# R^2 = 0.06726

# Наивысшую зарплату получают женщины молодого возраста, живущие в городе, разведенные или которые никогда не были в браке, без высшего образования и перерабатывающие.

## Задача 4

Условие

Набор данных: <https://www.kaggle.com/arashnic/hr-analytics-job-change-of-data-scientists?select=aug_test.csv>

Тип классификатора: *LogisticRegression (логистическая регрессия)*

Классификация по столбцу: *Education level (Masters – класс 0, остальные уровни – класс 1)*

1. Обработайте набор данных набор данных, указанный во втором столбце таблицы 4.1, 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.
2. 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?

Решение:

1. Опишем данные в столбцах:
   1. enrollee\_id : уникальный идентификатор кандидата
   2. city: Код города
   3. city\_ development \_index : индекс развития города (в масштабе)
   4. gender: Пол кандидата
   5. relevent\_experience: Соответствующий опыт кандидата
   6. enrolled\_university: Тип зачисленных университетских курсов, если таковые имеются.
   7. education\_level: Уровень образования кандидата
   8. major\_discipline :Обучение основной дисциплине кандидата
   9. experience: Кандидатский общий стаж в годах
   10. company\_size: Количество сотрудников в компании текущего работодателя
   11. company\_type : Тип текущего работодателя
   12. lastnewjob: разница в годах между предыдущей работой и текущей работой
   13. training\_hours: завершенные часы обучения
   14. target: 0 - Не ищу смены работы, 1 - Ищу смену работы
2. Смотрим информацию о датасете:

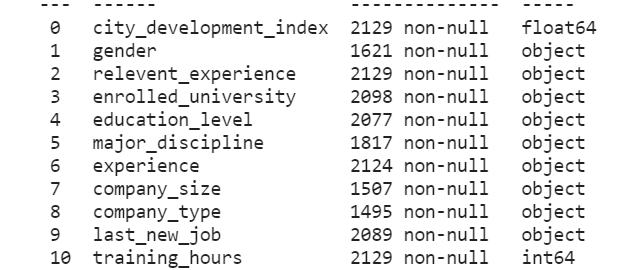


Рисунок 6. Информация о датасете.

Из Рисунка 1 видим, что некоторые признаки имеют пропуски и являются типами “object”, а не численными.

1. Обработаем данные для решения задачи классификации.
   1. Кодируем столбцы из категориального признака в численный. Смотри приложение 5 строки 28-61.
   2. Будем заполнять пропуски в данных значением по умолчанию – индексом 0. Смотри приложение 5 строки 62-66.
2. Выделяем целевой признак *- education\_level.* И удаляем его из данных, на основе которых будет обучаться классификатор. Смотри приложение 5 строки 67-68.
3. Разделяем набор данных на тестовую и обучающую выборки. Смотри приложение 5 строки 71-74.
4. Построим классификатор *LogisticRegression* сначала для обучающей выборки, а после для тестового набора данных*, используя оценщик GridSearchCV.* Смотри приложение 5 строки 76-83.



Рисунок 7. Оптимальные параметры, подобранные GridSearchCV.

* 1. Оценим точность построенного классификатора с помощью метрик precision, recall и F1 для тестового набора данных.

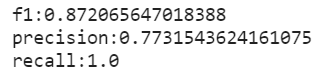


Рисунок 8. Результаты метрик для обучающей выборки.

* 1. Оценим точность построенного классификатора с помощью метрик precision, recall и F1 для обучающей выборки.

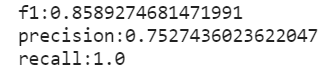


Рисунок 9. Результат метрик для тестового набора данных.

1. Построим классификатор типа Случайный Лес (Random Forest) для решения той же задачи классификации. Смотри приложение 5.
   1. Оценим качество с помощью метрик precision, recall и F1 на обучающей выборкe.

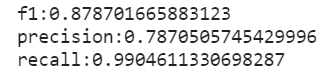


Рисунок 10. Результат метрик для обучающей выборки, классификатором Random Forest.

* 1. Оценим качество с помощью метрик precision, recall и F1 на тестовой выборкe.

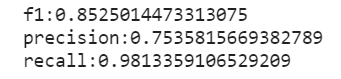


Рисунок 11. Результат метрик для тестовой выборки, классификатором Random Forest.

Полный код решения задачи приведён в Приложении 5.

Вывод:

На основе всех полученных данных я оцениваю классификатор Логистической регрессии, как довольно точный классификатор, так как он имеет довольно хорошую точность в 75%, полноту 100%, и f-меру в 86%.

Сравнивая результаты метрик классификаторов: Логистической регрессии и Рандомного леса, видим, что классификатор Логистической регрессии лучше, чем классификатор Рандомного леса. Смотри таблицу 2.

|  |  |  |
| --- | --- | --- |
| Тип метрики: | LogisticRegression | Random Forest |
| f1 | 85,9% | 85,2% |
| precision | 75,2% | 75,3 |
| recall | 100% | 98% |

Таблица 2. Сравнение метрик классификаторов Логистической регрессии и Рандомного леса.

Приложение 5

Устанавливаем необходимые библиотеки:

1. !pip install pandas
2. !pip install sklearn
3. import pandas as pd
4. import numpy as np
5. import warnings
6. warnings.filterwarnings('ignore')
7. data = pd.read\_csv('aug\_test.csv', index\_col='city')
8. #исключим столбец с уникальными значениями: enrollee\_id
9. data = data[data.columns.drop(['enrollee\_id'])]
10. data.head()
11. Описание столбцов:
12. enrollee\_id : уникальный идентификатор кандидата
13. city: Код города
14. city\_ development \_index : индекс развития города (в масштабе)
15. gender: Пол кандидата
16. relevent\_experience: Соответствующий опыт кандидата
17. enrolled\_university: Тип зачисленных университетских курсов, если таковые имеются.
18. education\_level: Уровень образования кандидата
19. major\_discipline :Обучение основной дисциплине кандидата
20. experience: Кандидатский общий стаж в годах
21. company\_size: Количество сотрудников в компании текущего работодателя
22. company\_type : Тип текущего работодателя
23. lastnewjob: разница в годах между предыдущей работой и текущей работой
24. training\_hours: завершенные часы обучения
25. target: 0 - Не ищу смены работы, 1 - Ищу смену работы
26. data.info()
27. #Кодируем столбец из категориального признака
28. data['gender'] = np.where(data['gender'] == 'Male', 1, 0)
29. data['education\_level'] = np.where(data['education\_level'] == 'Masters', 0, 1)
30. data['relevent\_experience'] = np.where(data['relevent\_experience'] == 'No relevent experience',0,1)
31. my\_set = set(data.enrolled\_university)
32. i = 0
33. for item in my\_set:
34. data['enrolled\_university'] = data['enrolled\_university'].replace(item, i)
35. i = i + 1
36. my\_set = set(data.last\_new\_job)
37. i = 0
38. for item in my\_set:
39. data['last\_new\_job'] = data['last\_new\_job'].replace(item, i)
40. i = i + 1
41. my\_set = set(data.major\_discipline)
42. i = 0
43. for item in my\_set:
44. data['major\_discipline'] = data['major\_discipline'].replace(item, i)
45. i = i + 1
47. my\_set = set(data.company\_type)
48. i = 0
49. for item in my\_set:
50. data['company\_type'] = data['company\_type'].replace(item, i)
51. i = i + 1
52. my\_set = set(data.company\_size)
53. i = 0
54. for item in my\_set:
55. data['company\_size'] = data['company\_size'].replace(item, i)
56. i = i + 1
57. my\_set = set(data.experience)
58. i = 0
59. for item in my\_set:
60. data['experience'] = data['experience'].replace(item, i)
61. i = i + 1
62. data.loc[data.experience == 'NaN' , 'experience'] = 0
63. data.loc[data.last\_new\_job == 'never' , 'last\_new\_job'] = 0
64. data.loc[data.last\_new\_job == 'NaN' , 'last\_new\_job'] = 0
65. data.loc[data.last\_new\_job == '>4' , 'last\_new\_job'] = 5
66. data.loc[data.gender == 'NaN' , 'gender'] = 0
67. target = data.education\_level
68. train = data.drop('education\_level', axis = 1)
69. train.info()
70. #y - целевая переменнная (target)
71. from sklearn.model\_selection import train\_test\_split
72. X\_train, X\_test, y\_train, y\_test = train\_test\_split(train, target, test\_size = 0.3, random\_state = 42)
73. N\_train, \_ = X\_train.shape
74. N\_test, \_ = X\_test.shape
75. print (N\_train, N\_test)
76. from sklearn.linear\_model import LogisticRegression
77. from sklearn.model\_selection import GridSearchCV
78. from sklearn.model\_selection import StratifiedKFold
79. grid={"C":np.logspace(-5,5,11), "penalty":["l1","l2"]}
80. logreg=LogisticRegression(solver='liblinear')
81. logreg\_cv=GridSearchCV(logreg,grid,cv=10)
82. logreg\_cv.fit(X\_train,y\_train)
83. logreg\_cv.best\_estimator\_
84. from sklearn.model\_selection import cross\_val\_score
85. print("f1:"+str(np.average(cross\_val\_score(logreg\_cv, X\_train, y\_train, scoring='f1'))))
86. print("precision:"+str(np.average(cross\_val\_score(logreg\_cv, X\_train, y\_train, scoring='precision'))))
87. print("recall:"+str(np.average(cross\_val\_score(logreg\_cv, X\_train, y\_train, scoring='recall'))))
88. from sklearn.model\_selection import cross\_val\_score
89. print("f1:"+str(np.average(cross\_val\_score(logreg\_cv, X\_test, y\_test, scoring='f1'))))
90. print("precision:"+str(np.average(cross\_val\_score(logreg\_cv, X\_test, y\_test, scoring='precision'))))
91. print("recall:"+str(np.average(cross\_val\_score(logreg\_cv, X\_test, y\_test, scoring='recall'))))
92. from sklearn.ensemble import RandomForestClassifier
93. param\_grid = { 'n\_estimators': [50, 100, 150],'max\_features': ['auto'],'max\_depth' : list(range(1, 10)), 'criterion' :['gini']}
94. RandForCrit = GridSearchCV(estimator = RandomForestClassifier(), param\_grid = param\_grid, cv = 5, refit = True)
95. RandForCrit.fit(X\_train, y\_train)
96. RandForCrit.predict(X\_test)
97. print("f1:"+str(np.average(cross\_val\_score(RandForCrit.best\_estimator\_, X\_train, y\_train, scoring='f1'))))
98. print("precision:"+str(np.average(cross\_val\_score(RandForCrit.best\_estimator\_, X\_train, y\_train, scoring='precision'))))
99. print("recall:"+str(np.average(cross\_val\_score(RandForCrit.best\_estimator\_, X\_train, y\_train, scoring='recall'))))
100. print("f1:"+str(np.average(cross\_val\_score(RandForCrit.best\_estimator\_, X\_test, y\_test, scoring='f1'))))
101. print("precision:"+str(np.average(cross\_val\_score(RandForCrit.best\_estimator\_, X\_test, y\_test, scoring='precision'))))
102. print("recall:"+str(np.average(cross\_val\_score(RandForCrit.best\_estimator\_, X\_test, y\_test, scoring='recall'))))

## Задача 5

Условие

Набор данных:  <https://www.kaggle.com/new-york-city/ny-2015-street-tree-census-tree-data?select=2015-street-tree-census-tree-data.csv>

Задача: Провести анализ датасета и сделать обработку данных. Также ответить на следующие вопросы:

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

оно есть.

2 Сколько категориальных признаков, какие?

3 Столбец с макимальным количеством уникальных значений категориального

признака?

4 Есть ли бинарные признаки?

5 Какие числовые признаки?

6 Есть ли пропуски?

7 Сколько объектов с пропусками?

8 Столбец с максимальным количеством пропусков?

9 Есть ли на ваш взгляд выбросы, аномальные значения?

10 Столбец с максимальным средним значением после нормировки признаков через

стандартное отклонение?

11 Столбец с целевым признаком?

12 Сколько объектов попадает в тренировочную выборку при использовании

train\_test\_split с параметрами test\_size = 0.3, random\_state = 42?

13 Между какими признаками наблюдается линейная зависимость (корреляция)?

14 Сколько признаков достаточно для объяснения 90% дисперсии после применения

метода PCA?

15 Какой признак вносит наибольший вклад в первую компоненту?

Решение:

1. Отвечаем на вопрос 1 - Сколько в датасете объектов и признаков? Дать описание каждому признаку, если оно есть.
   1. 1)tree\_id - уникальный идентификационный номер дерева (целое число)
   2. 2)block\_id - Идентификационный номер блок-фасада, на который отображено дерево (целое число)
   3. 3)created\_at - Дата нанесения деревьев на карту (дата)
   4. 4)tree\_dbh - Диаметр на высоте груди дерева (целое число)
   5. 5)stump\_diam - Диаметр пня (целое число)
   6. 6)curb\_loc - Расположено ли дерево вдоль бордюра или со смещением от него (Текст Значения домена: OnCurb OffsetFromCurb)
   7. 7)status - Статус дерева (Текст Доменные значения: Живой, Пень, Мертвый)
   8. 8)health - Здоровье деревьев (Текст Доменные ценности: Хорошо, Неплохо, Плохо)
   9. 9)spc\_latin - Научное / латинское название вида дерева (Текст)
   10. 10)spc\_common - Общее название породы деревьев (Текст)
   11. 11)steward - Количество замеченных признаков управления (Текст Значения доменов: 1 или 2 ,3 или 4, 4 или больше, Ничего)
   12. 12)guards - Наличие и тип охраны деревьев (Текст Доменные ценности: Вредные Полезно Нет Не уверен)
   13. 13)sidewalk - Повреждение тротуара в непосредственной близости от дерева (Текст Значения домена: Повреждение, Не повреждено)
   14. 14)user\_type - Категория пользователя, который собрал эту точку дерева (Текст Доменные ценности: Волонтер, TreesCount Персонал, Персонал NYC Parks)
   15. 15)root\_stone - Проблемы с корнями, вызванные брусчаткой в ложе дерева (Текст Значения домена: Да Нет)
   16. 16)root\_grate - Проблемы с корнями, вызванные металлическими решетками (Текст Значения домена: Да Нет)
   17. 17)root\_other - Наличие других проблем с корнями (Текст Значения домена: Да Нет)
   18. 18)trunk\_wire - Проблемы со стволом, вызванные тросом или проводами (Текст Значения домена: Да Нет)
   19. 19)trnk\_light - Проблемы со стволом, вызванные освещением (Текст Значения домена: Да Нет)
   20. 20)trnk\_other - Наличие других проблем со стволом (Текст Значения домена: Да Нет)
   21. 21)brch\_light - Проблемы с ветками, вызванные освещением или проводами (Текст Значения домена: Да Нет)
   22. 22)brch\_shoe - Проблемы с ветвями, вызванные обувью (Текст Значения домена: Да Нет)
   23. 23)brch\_other - Наличие других проблем с ветвями (Текст Значения домена: Да Нет)
   24. 24)address - Предполагаемый уличный адрес дерева (Текст)
   25. 25)zipcode - Почтовый индекс (целое число)
   26. 26)zip\_city - Город, полученный из почтового индекса (Текст)
   27. 27)cb\_num - Общественный совет (Целое число)
   28. 28)borocode - Геокодирование на основе местоположения точки дерева (Целое число Доменные значения: 1 (Манхэттен) 2 (Бронкс) 3 (Бруклин) 4 (Квинс) 5 (Статен-Айленд))
   29. 29)boroname - Геокодирование на основе местоположения точки дерева (Текст Доменные ценности: Манхэттен Бронкс, Бруклин Квинс, Статен-Айленд)
   30. 30)cncldist - Округ совета Нью-Йорка (Целое число)
   31. 31)st\_assem - Округ Ассамблеи штата Нью-Йорк (Целое число)
   32. 32)st\_senate - Сенатский округ штата Нью-Йорк (Целое число)
   33. 33)nta - Код района табуляции соседей (Текст)
   34. 34)nta\_name - Название зоны табуляции соседей (Текст)
   35. 35)boro\_ct - Переписной участок (Текст)
   36. 36)state - Государство (Текст)
   37. 37)latitude - Широта точки дерева (Двойной)
   38. 38)longitude - Долгота точки дерева (Двойной)
   39. 39)x\_sp - X координата точки дерева (Двойной)
   40. 40)y\_sp - Y координата точки дерева (Двойной)
2. ВОПРОС 2. Сколько категориальных признаков, какие?

Всего 9 категориальных признаков:

1. status Статус дерева
2. health Здоровье деревьев
3. steward Количество замеченных признаков управления
4. guards Наличие и тип охраны деревьев
5. user\_type Категория пользователя, который собрал эту точку дерева
6. borocode Геокодирование на основе местоположения точки дерева
7. boroname Геокодирование на основе местоположения точки дерева
8. ВОПРОС 3. Столбец с макимальным количеством уникальных значений категориального признака?

Максимальное число категориальных признаков у столбца borocode и borough

1. ВОПРОС 4. Есть ли бинарные признаки?

Да, есть:

1. curb\_loc
2. sidewalk
3. root\_stone
4. root\_grate
5. root\_other
6. trunk\_wire
7. trnk\_light
8. trnk\_other
9. brch\_light
10. brch\_shoe
11. brch\_other
12. ВОПРОС 5. Какие числовые признаки?
    1. tree\_id
    2. block\_id
    3. created\_at
    4. tree\_dbh
    5. stump\_diam
    6. zipcode
    7. cb\_num
    8. cncldist
    9. st\_assem
    10. st\_senate
    11. latitude
    12. longitude
    13. x\_sp
    14. y\_sp
13. Исключим столбцы с уникальными значениями и включающие результат других столбцов: tree\_id, created\_at, address, problems.
14. ВОПРОС 6. Есть ли пропуски?
    1. Воспользуемся командой info() , чтобы увидеть общую информацию о датасете.

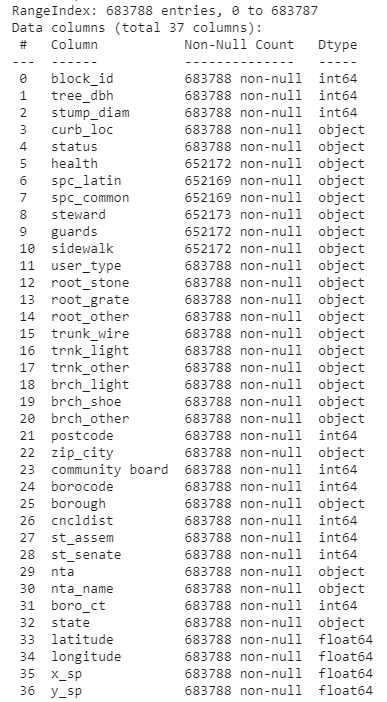


Рисунок 12. Информация о датасете.

Как можем наблюдать из Рисунка 12 – у нас есть пропуски.

1. ВОПРОС 7. Сколько объектов с пропусками?

Основываясь на результате info() - видим, что у 6-ти объектов есть пропуски.

1. ВОПРОС 8. Столбец с максимальным количеством пропусков?

Воспользуемся командой isnull(), для подсчета максимального числа пропусков.

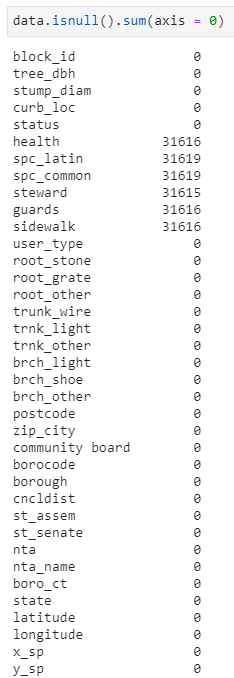


Рисунок 13. Результат работы isnull()

Следовательно, столбец/столбцы с наибольшим/им числом/ами пропусков - это spc\_latin = 31619 и spc\_common = 31619

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

Аномальных значений и выбросов не замечено. Смотри приложение 6.

1. ВОПРОС 10. Столбец с максимальным средним значением после нормировки признаков через стандартное отклонение? 

Рисунок 14. Поиск столбца с максимальным средним значением после нормировки признаков, через стандартное отклонение.

1. ВОПРОС 11. Столбец с целевым признаком?

Так как, мне надо выделить районы, в которых состояние деревьев аномально хорошее или плохое; выделить наилучший район по состоянию деревьев; выделить виды деревьев (или более общие элементы классификации), обладающие наилучшими показателями «здоровья», то будем оценивать столбцы:

1. 1)status - Указывает, является ли дерево живым, стоящим мертвым или пнем.
2. zip\_city - Город, полученный из почтового индекса
3. 3)root\_stone - Проблемы с корнями, вызванные брусчаткой в ложе дерева
4. 4)root\_grate - Проблемы с корнями, вызванные металлическими решетками
5. 5)root\_other - Наличие других проблем с корнями
6. 6)trunk\_wire - Проблемы со стволом, вызванные тросом или проводами
7. 7)trnk\_light - Проблемы со стволом, вызванные освещением
8. 8)trnk\_other - Наличие других проблем со стволом
9. 9)brch\_light - Проблемы с ветками, вызванные освещением или проводами
10. 10)brch\_shoe - Проблемы с ветвями, вызванные обувью
11. 11)brch\_other - Наличие других проблем с ветвями
12. 12)spc\_common - Общее название породы деревьев
13. Обрабатываем категориальные и бинарные признаки датасета. Смотри приложение 6.
14. ВОПРОС 12. Сколько объектов попадает в тренировочную выборку при использовании train\_test\_split с параметрами test\_size = 0.3, random\_state = 42?



Рисунок 15. Исключаем столбце с целевыми признаками из датасета для тренировки.

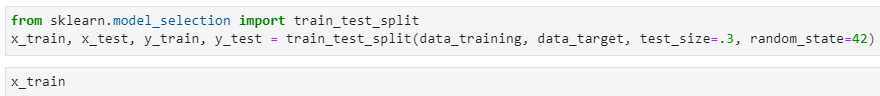


Рисунок 16. Разделение данных на тренировочные и тестовые.

В тренировочную выборку попадет 478651 объектов.

1. ВОПРОС 13. Между какими признаками наблюдается линейная зависимость (корреляция)?

Проверим коэффициент VIF (Variable Inflation Factors) для всех признаков. Если VIF > 10, то есть четкая линейная зависимость.

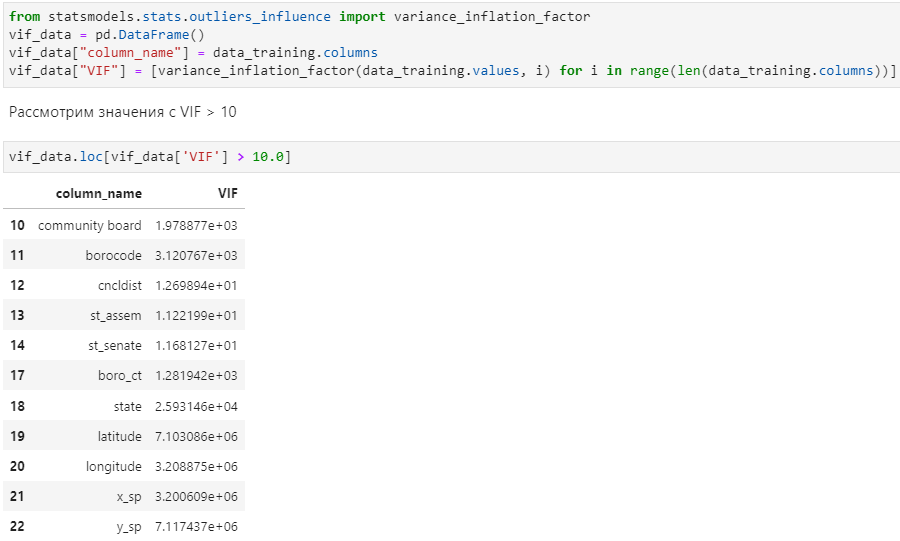


Рисунок 17. Поиск признаков с коэффициентов VIF больше 10.

1. ВОПРОС 14. Сколько признаков достаточно для объяснения 90% дисперсии после применения метода PCA?

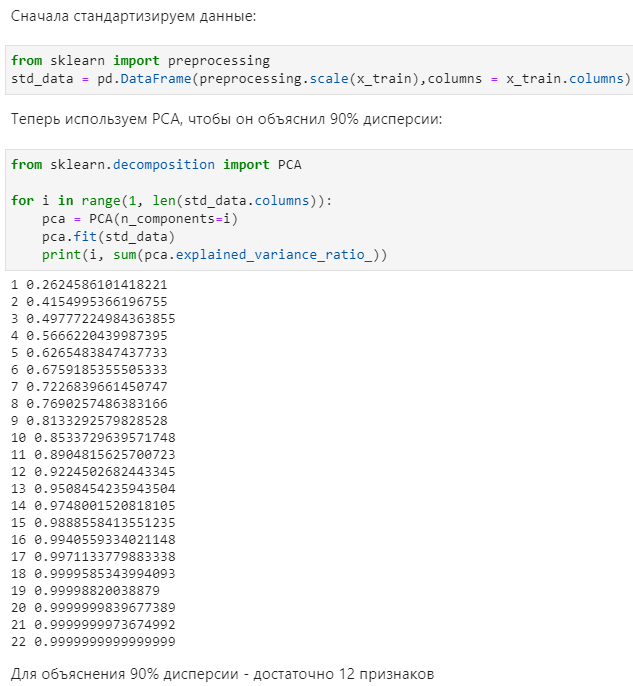


Рисунок 18. Применение метода главных компонентов.

1. Вопрос 15. Какой признак вносит наибольший вклад в первую компоненту?

Наибольший вклад в первую компоненту вносит 11-ый признак - borocode. Смотри приложение 6.

Приложение 6.

1. !pip install pandas
2. !pip install sklearn
3. import pandas as pd
4. import numpy as np
5. import warnings
6. warnings.filterwarnings('ignore')
7. data = pd.read\_csv('2015-street-tree-census-tree-data.csv')
8. #исключим столбец с уникальными значениями: tree\_id
9. data = data.drop('tree\_id', axis = 1)
10. #исключим столбец, включающий результаты других столбцов
11. data = data.drop('problems', axis = 1)
12. #исключим столбец с индивидуальными параметрами
13. data = data.drop('created\_at', axis = 1)
14. #исключим столбец с индивидуальными параметрами
15. data = data.drop('address', axis = 1)
16. #исключим столбцы с признаками, которые отсутствуют в документации
17. data = data.drop('council district', axis = 1)
18. data = data.drop('census tract', axis = 1)
19. data = data.drop('bin', axis = 1)
20. data = data.drop('bbl', axis = 1)
21. ctgrl\_priz = ['status','health','steward','guards','user\_type','borocode','borough']
22. for column in data.columns:
23. if column in ctgrl\_priz:
24. print(column, data[column].nunique())
25. data.info()
26. data.isnull().sum(axis = 0)
27. for column in data.columns:
28. print(data[column].value\_counts())
29. import math
30. arr = ['block\_id','tree\_dbh','stump\_diam','postcode','community board','cncldist','st\_assem','st\_senate','boro\_ct','latitude','longitude','x\_sp','y\_sp']
31. for column in arr:
32. data[column] = (data[column] - data[column].mean())/(math.sqrt(data[column].var()))
33. avrg = -math.inf
34. for column in arr:
35. if data[column].mean() > avrg:
36. avrg = data[column].mean()
37. print(avrg, column)
38. data['state'] = np.where(data['state'] == 'New York', 1, 0)
39. data.loc[data.spc\_latin == 'NaN', 'spc\_latin'] = 0
40. data.loc[data.spc\_common == 'NaN', 'spc\_common'] = 0
41. data.loc[data.steward == 'NaN', 'steward'] = 0
42. data.loc[data.guards == 'NaN', 'guards'] = 0
43. data.loc[data.sidewalk == 'NaN', 'sidewalk'] = 0
44. #Расположено ли дерево вдоль бордюра или со смещением от него
45. data['curb\_loc'] = np.where(data['curb\_loc'] == 'OnCurb', 1, 0)
46. #Повреждение тратуара в непосредственной близости от дерева
47. data['sidewalk'] = np.where(data['sidewalk'] == 'Damage', 1, 0)
48. #Указывает на наличие проблемы с корнями, вызванной брусчаткой в ложе дерева
49. data['root\_stone'] = np.where(data['root\_stone'] == 'Yes', 1, 0)
50. #Указывает на наличие проблемы с корнями, вызванной металлическими решетками в ложе дерева
51. data['root\_grate'] = np.where(data['root\_grate'] == 'Yes', 1, 0)
52. #Указывает на наличие других корневых проблем
53. data['root\_other'] = np.where(data['root\_other'] == 'Yes', 1, 0)
54. #Указывает на наличие проблемы со стволом, вызванной проволокой или веревкой, обмотанной вокруг ствола
55. data['trunk\_wire'] = np.where(data['trunk\_wire'] == 'Yes', 1, 0)
56. #Указывает на наличие проблем со стволом, вызванных установленным на дереве освещением
57. data['trnk\_light'] = np.where(data['trnk\_light'] == 'Yes', 1, 0)
58. #Указывает на наличие других проблем с магистралью
59. data['trnk\_other'] = np.where(data['trnk\_other'] == 'Yes', 1, 0)
60. #Указывает на наличие проблемы с ветвями, вызванной светильниками (обычно струнными) или проводами в ветвях
61. data['brch\_light'] = np.where(data['brch\_light'] == 'Yes', 1, 0)
62. #Указывает на наличие проблемы с ветками, вызванной кроссовками в ветках
63. data['brch\_shoe'] = np.where(data['brch\_shoe'] == 'Yes', 1, 0)
64. #Указывает на наличие других проблем с ветвями
65. data['brch\_other'] = np.where(data['brch\_other'] == 'Yes', 1, 0)
66. #Город, полученный из почтового индекса
67. my\_set = set(data.zip\_city)
68. i = 0
69. for item in my\_set:
70. data['zip\_city'] = data['zip\_city'].replace(item, i)
71. i = i + 1
72. #Это название NTA, соответствующее области табуляции района по переписи населения США 2010 года, в которую попадает точка дерева
73. my\_set = set(data.nta\_name)
74. i = 0
75. for item in my\_set:
76. data['nta\_name'] = data['nta\_name'].replace(item, i)
77. i = i + 1
78. #Это код NTA, соответствующий району, в который попадает точка дерева, согласно переписи населения США 2010 года.
79. my\_set = set(data.nta)
80. i = 0
81. for item in my\_set:
82. data['nta'] = data['nta'].replace(item, i)
83. i = i + 1
84. #Название района, в котором находится точка произрастания деревьев
85. my\_set = set(data.borough)
86. i = 0
87. for item in my\_set:
88. data['borough'] = data['borough'].replace(item, i)
89. i = i + 1
90. #Город, полученный из почтового индекса. Часто (но не всегда) это то же самое, что и район.
91. my\_set = set(data.postcode)
92. i = 0
93. for item in my\_set:
94. data['postcode'] = data['postcode'].replace(item, i)
95. i = i + 1
96. #Указывает, присутствует ли ограждение, и считает ли пользователь, что это полезное или вредное ограждение. Не регистрируется для мертвых деревьев и пней.
97. my\_set = set(data.guards)
98. i = 0
99. for item in my\_set:
100. data['guards'] = data['guards'].replace(item, i)
101. i = i + 1
102. #Это поле описывает категорию пользователя, который собрал данные этой точки дерева.
103. my\_set = set(data.user\_type)
104. i = 0
105. for item in my\_set:
106. data['user\_type'] = data['user\_type'].replace(item, i)
107. i = i + 1
109. #Указывает количество уникальных признаков бережного отношения, отмеченных для данного дерева. Не регистрируется для пней или мертвых деревьев.
110. my\_set = set(data.steward)
111. i = 0
112. for item in my\_set:
113. data['steward'] = data['steward'].replace(item, i)
114. i = i + 1
115. #Общее название вида, например, "красный клен"
116. my\_set = set(data.spc\_common)
117. i = 0
118. for item in my\_set:
119. data['spc\_common'] = data['spc\_common'].replace(item, i)
120. i = i + 1
122. #Научное название вида, например, "Acer rubrum".
123. my\_set = set(data.spc\_latin)
124. i = 0
125. for item in my\_set:
126. data['spc\_latin'] = data['spc\_latin'].replace(item, i)
127. i = i + 1
128. #Указывает, является ли дерево живым, стоящим мертвым или пнем.
129. data.loc[data.status == 'Alive', 'status'] = 2
130. data.loc[data.status == 'Stump', 'status'] = 1
131. data.loc[data.status == 'Dead', 'status'] = 0
132. #Здоровье деревьев
133. data.loc[data.health == 'Good', 'health'] = 2
134. data.loc[data.health == 'Fair', 'health'] = 1
135. data.loc[data.health == 'Poor', 'health'] = 0
136. data.loc[data.health == 'NaN', 'health'] = 0
137. data\_training = data.drop(['health',
138. 'borough',
139. 'root\_stone',
140. 'root\_grate',
141. 'root\_other',
142. 'trunk\_wire',
143. 'trnk\_light',
144. 'trnk\_other',
145. 'brch\_light',
146. 'brch\_shoe',
147. 'zip\_city',
148. 'brch\_other',
149. 'status',
150. 'spc\_common'],
151. axis = 1)
152. data\_target = data[['health',
153. 'borough',
154. 'root\_stone',
155. 'root\_grate',
156. 'root\_other',
157. 'trunk\_wire',
158. 'trnk\_light',
159. 'trnk\_other',
160. 'brch\_light',
161. 'brch\_shoe',
162. 'zip\_city',
163. 'brch\_other',
164. 'status',
165. 'spc\_common']]
166. data\_training.info()
167. from sklearn.model\_selection import train\_test\_split
168. x\_train, x\_test, y\_train, y\_test = train\_test\_split(data\_training, data\_target, test\_size=.3, random\_state=42)
169. x\_train
170. from statsmodels.stats.outliers\_influence import variance\_inflation\_factor
171. vif\_data = pd.DataFrame()
172. vif\_data["column\_name"] = data\_training.columns
173. vif\_data["VIF"] = [variance\_inflation\_factor(data\_training.values, i) for i in range(len(data\_training.columns))]
174. vif\_data.loc[vif\_data['VIF'] > 10.0]
175. from sklearn import preprocessing
176. std\_data = pd.DataFrame(preprocessing.scale(x\_train),columns = x\_train.columns)
177. from sklearn.decomposition import PCA
178. for i in range(1, len(std\_data.columns)):
179. pca = PCA(n\_components=i)
180. pca.fit(std\_data)
181. print(i, sum(pca.explained\_variance\_ratio\_))
182. pca = PCA(n\_components=12)
183. x = pca.fit(std\_data)
184. first\_component = pca.components\_[0]
185. i = 0
186. answ = first\_component[i]
187. for j in range(0, len(first\_component)):
188. if abs(first\_component[j]) > abs(answ):
189. answ = first\_component[j]
190. i = j
191. print(i, answ)
192. std\_data.iloc[:, 11]

## Задача 6

Набор данных:  <https://www.kaggle.com/new-york-city/ny-2015-street-tree-census-tree-data?select=2015-street-tree-census-tree-data.csv>

Задачи:

1. выделить районы, в которых состояние деревьев аномально хорошее или плохое;
2. выделить наилучший район по состоянию деревьев;
3. выделить виды деревьев (или более общие элементы классификации), обладающие наилучшими показателями «здоровья»

Решение:

1. Считываем набор данных и обрабатываем его. Смотри строки из Приложения 7: 1-157.
2. Опишем каждый признак. Смотри пункт 2 из задачи 5.
3. Задача 1. Выделить районы, в которых состояние деревьев аномально хорошее или плохое.
   1. Создадим новый признак “problems”, состоящий из суммы положительных ответов о проблемах деревьев. Смотри рисунок 19.
   2. Построим пару графиков для поиска аномальных состояний деревьев. Смотри рисунок 19.

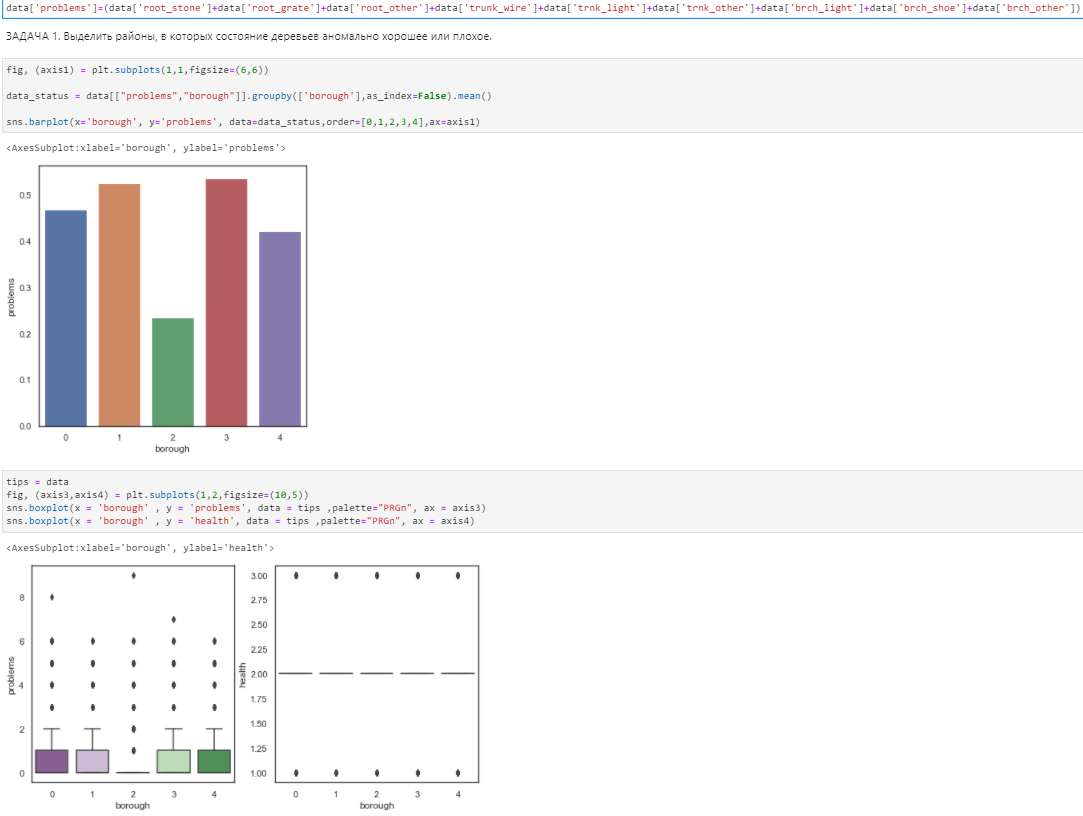


Рисунок 19. Создание нового признака [‘problems’] и 3-ёх графиков для поиска аномальных значений.

Проанализировав графики можно прийти к выводу, что деревья в районе под номером 2 - Staten Island имеют аномально хорошее состояние.

1. ЗАДАЧА 2. Выделить наилучший район по состоянию деревьев.

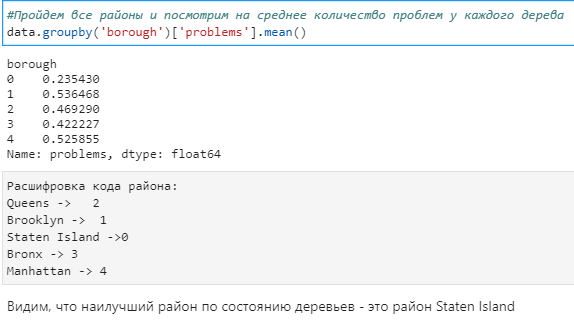


Рисунок 20. Поиск района с наименьшим количеством проблем у деревьев.

Из данных с рисунка 2 видим, что район с наилучшим состоянием деревьев – это Staten Island.

1. ЗАДАЧА 3. Выделить виды деревьев (или более общие элементы классификации), обладающие наилучшими показателями «здоровья»

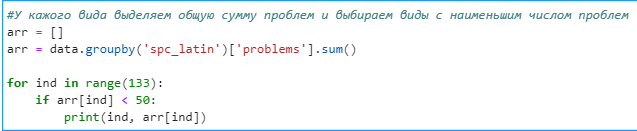


Рисунок 21. Суммируем число проблем у каждого вида дерева. Элемент классифицируется по его названию на Латыни.

Проанализировав результат фрагмента кода из рисунка 3 имеем, что виды деревья с наилучшим состоянием здоровья – это деревья:

1. Tilia cordata
2. Cotinus coggygria
3. Quercus alba
4. Ulmus pumila
5. Acer buergerianum
6. Acer nigrum
7. Pinus resinosa
8. Quercus rubra
9. Cornus florida
10. Carpinus betulus
11. Cedrus atlantica
12. Quercus velutina
13. Liquidambar styraciflua
14. Prunus virginiana
15. Picea
16. Aesculus glabra
17. Fagus grandifolia
18. Cornus mas
19. Magnolia grandiflora
20. Carya glabra
21. Acer tataricum
22. Syringa reticulata
23. Carpinus caroliniana
24. Chamaecyparis pisifera
25. Carpinus japonica
26. Fraxinus pennsylvanica
27. Pyrus calleryana
28. Crataegus
29. Acer negundo
30. Tsuga canadensis
31. Parrotia persica
32. Acer ginnala
33. Pinus

Вывод:

Основываясь на результатах из пунктов 3,4,5 мы приходим к выводу, что район с аномально хорошим и наилучшим состоянием деревьев – это район Staten Island; виды деревьев с наилучшими показателями здоровья - Смотри вывод из пункта 5.

Приложение 7:

1. import pandas as pd
2. import numpy as np
3. import matplotlib.pyplot as plt # библиотека для визуализации
4. import seaborn as sns
5. sns.set(style="white", color\_codes=True)
6. %matplotlib inline
7. import warnings
8. warnings.filterwarnings('ignore')
9. data = pd.read\_csv('2015-street-tree-census-tree-data.csv')
10. import math
11. arr = ['block\_id','tree\_dbh','stump\_diam','postcode','community board','cncldist','st\_assem','st\_senate','latitude','longitude','x\_sp','y\_sp']
12. for column in arr:
13. data[column] = (data[column] - data[column].mean())/(math.sqrt(data[column].var()))
15. data['state'] = np.where(data['state'] == 'New York', 1, 0)
16. #Расположено ли дерево вдоль бордюра или со смещением от него
17. data['curb\_loc'] = np.where(data['curb\_loc'] == 'OnCurb', 1, 0)
18. #Повреждение тратуара в непосредственной близости от дерева
19. data['sidewalk'] = np.where(data['sidewalk'] == 'Damage', 1, 0)
20. #Указывает на наличие проблемы с корнями, вызванной брусчаткой в ложе дерева
21. data['root\_stone'] = np.where(data['root\_stone'] == 'Yes', 1, 0)
22. #Указывает на наличие проблемы с корнями, вызванной металлическими решетками в ложе дерева
23. data['root\_grate'] = np.where(data['root\_grate'] == 'Yes', 1, 0)
24. #Указывает на наличие других корневых проблем
25. data['root\_other'] = np.where(data['root\_other'] == 'Yes', 1, 0)
26. #Указывает на наличие проблемы со стволом, вызванной проволокой или веревкой, обмотанной вокруг ствола
27. data['trunk\_wire'] = np.where(data['trunk\_wire'] == 'Yes', 1, 0)
28. #Указывает на наличие проблем со стволом, вызванных установленным на дереве освещением
29. data['trnk\_light'] = np.where(data['trnk\_light'] == 'Yes', 1, 0)
30. #Указывает на наличие других проблем с магистралью
31. data['trnk\_other'] = np.where(data['trnk\_other'] == 'Yes', 1, 0)
32. #Указывает на наличие проблемы с ветвями, вызванной светильниками (обычно струнными) или проводами в ветвях
33. data['brch\_light'] = np.where(data['brch\_light'] == 'Yes', 1, 0)
34. #Указывает на наличие проблемы с ветками, вызванной кроссовками в ветках
35. data['brch\_shoe'] = np.where(data['brch\_shoe'] == 'Yes', 1, 0)
36. #Указывает на наличие других проблем с ветвями
37. data['brch\_other'] = np.where(data['brch\_other'] == 'Yes', 1, 0)
38. #Город, полученный из почтового индекса
39. my\_set = set(data.zip\_city)
40. i = 0
41. for item in my\_set:
42. data['zip\_city'] = data['zip\_city'].replace(item, i)
43. i = i + 1
44. #Это название NTA, соответствующее области табуляции района по переписи населения США 2010 года, в которую попадает точка дерева
45. my\_set = set(data.nta\_name)
46. i = 0
47. for item in my\_set:
48. data['nta\_name'] = data['nta\_name'].replace(item, i)
49. i = i + 1
50. #Это код NTA, соответствующий району, в который попадает точка дерева, согласно переписи населения США 2010 года.
51. my\_set = set(data.nta)
52. i = 0
53. for item in my\_set:
54. data['nta'] = data['nta'].replace(item, i)
55. i = i + 1
56. #Название района, в котором находится точка произрастания деревьев
57. my\_set = set(data.borough)
58. i = 0
59. for item in my\_set:
60. data['borough'] = data['borough'].replace(item, i)
61. i = i + 1
62. #Город, полученный из почтового индекса. Часто (но не всегда) это то же самое, что и район.
63. my\_set = set(data.postcode)
64. i = 0
65. for item in my\_set:
66. data['postcode'] = data['postcode'].replace(item, i)
67. i = i + 1
68. #Указывает, присутствует ли ограждение, и считает ли пользователь, что это полезное или вредное ограждение. Не регистрируется для мертвых деревьев и пней.
69. my\_set = set(data.guards)
70. i = 0
71. for item in my\_set:
72. data['guards'] = data['guards'].replace(item, i)
73. i = i + 1
74. #Это поле описывает категорию пользователя, который собрал данные этой точки дерева.
75. my\_set = set(data.user\_type)
76. i = 0
77. for item in my\_set:
78. data['user\_type'] = data['user\_type'].replace(item, i)
79. i = i + 1
81. #Указывает количество уникальных признаков бережного отношения, отмеченных для данного дерева. Не регистрируется для пней или мертвых деревьев.
82. my\_set = set(data.steward)
83. i = 0
84. for item in my\_set:
85. data['steward'] = data['steward'].replace(item, i)
86. i = i + 1
87. #Общее название вида, например, "красный клен"
88. my\_set = set(data.spc\_common)
89. i = 0
90. for item in my\_set:
91. data['spc\_common'] = data['spc\_common'].replace(item, i)
92. i = i + 1
94. #Научное название вида, например, "Acer rubrum".
95. my\_set = set(data.spc\_latin)
96. i = 0
97. for item in my\_set:
98. data['spc\_latin'] = data['spc\_latin'].replace(item, i)
99. i = i + 1
100. #Здоровье деревьев
101. my\_set = set(data.health)
102. i = 0
103. for item in my\_set:
104. data['health'] = data['health'].replace(item, i)
105. i = i + 1
107. data.loc[data.health == 'NaN', 'health'] = 0
108. data.loc[data.health == 0, 'health'] = 3
110. #Указывает, является ли дерево живым, стоящим мертвым или пнем.
111. data.loc[data.status == 'Alive', 'status'] = 2
112. data.loc[data.status == 'Stump', 'status'] = 1
113. data.loc[data.status == 'Dead', 'status'] = 0
114. data.loc[data.spc\_latin == 'NaN', 'spc\_latin'] = 0
115. data.loc[data.spc\_common == 'NaN', 'spc\_common'] = 0
116. data.loc[data.steward == 'NaN', 'steward'] = 0
117. data.loc[data.guards == 'NaN', 'guards'] = 0
118. data.loc[data.sidewalk == 'NaN', 'sidewalk'] = 0
119. #исключим столбец с уникальными значениями: tree\_id
120. data = data.drop('tree\_id', axis = 1)
121. #исключим столбец, включающий результаты других столбцов
122. data = data.drop('problems', axis = 1)
123. #исключим столбец с индивидуальными параметрами
124. data = data.drop('created\_at', axis = 1)
125. #исключим столбец с индивидуальными параметрами
126. data = data.drop('address', axis = 1)
127. #исключим столбцы с признаками, которые отсутствуют в документации
128. data = data.drop('council district', axis = 1)
129. data = data.drop('census tract', axis = 1)
130. data = data.drop('bin', axis = 1)
131. data = data.drop('bbl', axis = 1)
132. #Создадим новый признак - problems {тип: число}, состоит из суммы положительных ответов о проблемах корней
133. #Чем больше значение problems - тем состояние дерева хуже
134. data['problems']=(data['root\_stone']+data['root\_grate']+data['root\_other']+data['trunk\_wire']+data['trnk\_light']+data['trnk\_other']+data['brch\_light']+data['brch\_shoe']+data['brch\_other'])
135. fig, (axis1) = plt.subplots(1,1,figsize=(6,6))
136. data\_status = data[["problems","borough"]].groupby(['borough'],as\_index=False).mean()
137. sns.barplot(x='borough', y='problems', data=data\_status,order=[0,1,2,3,4],ax=axis1)
138. tips = data
139. fig, (axis3,axis4) = plt.subplots(1,2,figsize=(10,5))
140. sns.boxplot(x = 'borough' , y = 'problems', data = tips ,palette="PRGn", ax = axis3)
141. sns.boxplot(x = 'borough' , y = 'health', data = tips ,palette="PRGn", ax = axis4)
142. #Пройдем все районы и посмотрим на количство проблем у каждого дерева
143. data.groupby('borough')['problems'].sum()
144. #У кажого вида выделяем общую сумму проблем и выбираем виды с наименьшим числом проблем
145. arr = []
146. arr = data.groupby('spc\_latin')['problems'].sum()
147. for ind in range(133):
148. if arr[ind] < 50:
149. print(ind, arr[ind])