**Задание 3. Загрузка данных из Интернет (Web-scraping) средствами R и анализ данных в R.**

Выполнила студентка 2 курса

группы 09-715(1)

Анисимова Татьяна.

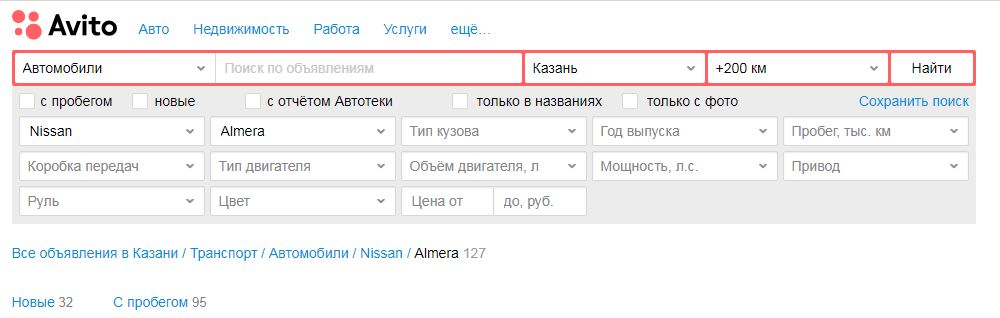
**Текст задания:**

Пусть перед нами стоит задача прогнозирования цены на товары (например, на автомобили) на основе объявлений, публикуемых на сайте **avito.ru** (или на подобных сайтах). Задайте параметры поиска так, чтобы в выборку попало более 50 однородных объектов. Выходным параметром для прогноза является цена. Выберите входные параметры для прогноза.

Далее полученные данные нужно прочитать в R, при необходимости преобразовать к нужному формату и построить модель для прогнозирования. В данной задаче для прогнозирования применим **линейную регрессию**. Можете использовать готовые функции из R или реализовать алгоритм линейной регрессии самостоятельно. Оцените качество построенной модели. Сделайте выводы.

**Описание выполненной работы:**

На сайте **avito.ru** были выбраны автомобили Nissan Almera, в выборку попало 127 автомобилей.



Для извлечения данных с сайта используем библиотеку **rvest**.

> library(rvest)

Url адрес сайта будем хранить в переменной nissan\_url:

> nissan\_url= 'https://www.avito.ru/kazan/avtomobili/nissan/almera?radius=200'

Воспользуемся функцией read\_html(), которая возвращает XML-документ, содержащий всю информацию о веб-странице:

> pages <- read\_html(nissan\_url)

> pages

{xml\_document}

<html>

[1] <head>\n<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">\n<script>\n window.dataLayer = [{"pageType":"SearchResults","q ...

[2] <body class=" " id="catalog">\n <noscript> <iframe src="//www.googletagmanager.com/ns.html?id=GTM-KP9Q9H" height="0" width="0" style ...

На сайте с объявлениями чаще всего бывает несколько страниц. Для начала найдем максимальный номер страницы.

> # функция для получения номера последней страницы

> get\_last\_page\_numb <- function(html){

+

+ pages\_data <- html %>%

+ html\_nodes('.pagination-page') %>%

+ html\_text()

+

+ pages\_data[(length(pages\_data)-1)] %>%

+ unname() %>%

+ as.numeric()

+ }

> # получим номер последней страницы

> (last\_page\_numb <- get\_last\_page\_numb(pages))

[1] 3

Таким образом, мы имеет 3 страницы с объявлениями о продаже автомобилей. Теперь получим список url адресов каждой страницы. Заметим, что url первой страницы имеет вид:

* https://www.avito.ru/kazan/avtomobili/nissan/almera?radius=200

Url последующих страниц имеет вид:

* [https://www.avito.ru/kazan/avtomobili/nissan/almera?p=**page\_numb**&radius=200](https://www.avito.ru/kazan/avtomobili/nissan/almera?p=page_numb&radius=200),

где page\_numb – номер страницы.

Подключим пакет **stringr**, чтобы использовать регулярные выражения:

> library(stringr)

> # получим список страниц

> list\_of\_pages = character();

> list\_of\_pages = append(list\_of\_pages, nissan\_url)

> list\_of\_pages = append(list\_of\_pages, str\_c(short\_url, paste0('p=', 2:last\_page\_numb, '&radius=200')));

Список url адресов каждой страницы:

> list\_of\_pages

[1] "https://www.avito.ru/kazan/avtomobili/nissan/almera?radius=200" "https://www.avito.ru/kazan/avtomobili/nissan/almera?p=2&radius=200"

[3] "https://www.avito.ru/kazan/avtomobili/nissan/almera?p=3&radius=200"

С помощью функции html\_nodes() мы можем обратиться к элементам определенного класса. Например, информация о каждом автомобиле храниться в классе item\_hide-elements, чтобы получить все элементы этого класса, необходимо вызвать функцию: html\_nodes(".item\_hide-elements").

С помощью функции html\_text() мы можем вытащить текстовую информацию из

html документа. Например, чтобы узнать год выпуска автомобиля, мы обращаемся к классу item-description-title, извлекаем текстовую информацию с помощью html\_text() и удаляем все символы переноса строки с помощью функции str\_trim().

+ #вытаскиваем год выпуска

+ tmp <- results %>% html\_nodes(".item-description-title")%>% html\_text()%>%str\_trim()

Используя регулярные выражения, получаем только числа:

+ year\_tmp = as.numeric(str\_extract(tmp, "[:digit:]+"));

Аналогичным образом, мы вытаскиваем информацию о цене (price) и характеристиках автомобиля (info).

Для того чтобы получить ссылки на подробную информацию об автомобиле, мы используем функцию html\_attr(). Например, вызвав html\_attr("href"), мы получим информацию об атрибуте **href.**

+ #вытаскиваем ccылки на подробную информацию об автомобиле

+ ref <- results %>% html\_nodes(".item-description-title-link") %>% html\_attr("href")

Однако ссылки имеют следующий вид:

> ref

[1] "/ulyanovsk/avtomobili/nissan\_almera\_2001\_1381613224"

"/kazan/avtomobili/nissan\_almera\_2016\_1080761872"

[3] "/kazan/avtomobili/nissan\_almera\_2005\_1338280987"

"/yoshkar-ola/avtomobili/nissan\_almera\_2005\_1319972476"

[5] "/cheboksary/avtomobili/nissan\_almera\_2005\_1249390937"

"/yoshkar-ola/avtomobili/nissan\_almera\_2000\_1549661214"

[7] "/yoshkar-ola/avtomobili/nissan\_almera\_2013\_1710196370"

"/ulyanovsk/avtomobili/nissan\_almera\_2013\_1138201954"

[9] "/cheboksary/avtomobili/nissan\_almera\_2014\_1590860184"

"/elabuga/avtomobili/nissan\_almera\_2015\_943579456"

[11] "/cheboksary/avtomobili/nissan\_almera\_2007\_1357176510"

"/yoshkar-ola/avtomobili/nissan\_almera\_2015\_1324525184"

[13] "/kazan/avtomobili/nissan\_almera\_2013\_966444730"

"/yoshkar-ola/avtomobili/nissan\_almera\_2001\_1623784007"

Чтобы получить url адрес необходимой страницы, добавим к полученным ссылкам

> main\_url = 'https://www.avito.ru'

Сложить строки можно с помощью функции paste0():

+ ref <- paste0(main\_url, ref)

Теперь ссылки имеют вид:

> ref

[1] "https://www.avito.ru/ulyanovsk/avtomobili/nissan\_almera\_2001\_1381613224"

[2] "https://www.avito.ru/kazan/avtomobili/nissan\_almera\_2016\_1080761872"

[3] "https://www.avito.ru/kazan/avtomobili/nissan\_almera\_2005\_1338280987"

[4] "https://www.avito.ru/yoshkar-ola/avtomobili/nissan\_almera\_2005\_1319972476"

[5] "https://www.avito.ru/cheboksary/avtomobili/nissan\_almera\_2005\_1249390937"

Таким образом, проходимся по каждой странице объявлений и вытаскиваем год выпуска, цену, характеристики автомобиля, ссылки на подробную информацию об автомобиле:

> for(i in 1:last\_page\_numb)

+ {

+ page <- read\_html(list\_of\_pages[i]);

+ results <- page %>% html\_nodes(".item\_hide-elements")

+

+ #вытаскиваем год выпуска

+ tmp <- results %>% html\_nodes(".item-description-title")%>% html\_text()%>%str\_trim()

+ year\_tmp = as.numeric(str\_extract(tmp, "[:digit:]+"));

+ year = append(year, year\_tmp);

+

+ #вытаскиваем цену автомобиля

+ price\_tmp <- results %>% html\_nodes(".about\_bold-price") %>% html\_nodes(".price") %>% html\_text()

+ price\_tmp = as.numeric(str\_replace\_all(price\_tmp, "[ \n\r\t\\\u20bd]+", ""))

+ price = append(price, price\_tmp);

+

+ #вытаскиваем информацию о характеристиках автомобиля

+ info\_tmp <- results %>% html\_nodes(".about\_bold-price")%>% html\_nodes(".specific-params\_block")%>% html\_text()%>%str\_trim()

+ info\_tmp = str\_replace\_all(info\_tmp, "[ ]+", "");

+ info = append(info, info\_tmp);

+

+ #вытаскиваем ccылки на подробную информацию об автомобиле

+ ref <- results %>% html\_nodes(".item-description-title-link") %>% html\_attr("href")

+ ref <- paste0(main\_url, ref)

+ ref\_info = append(ref\_info, ref)

+ }

> #год выпуска

> year

[1] 2013 2016 2013 2013 2006 2007 2015 2006 2016 2018 2015 2001 2013 2005 2013 2015 2005 2000 2013 2007 2015 2001 2013 2015 2014 2013 2013 2006

[29] 1997 2016 2000 2018 2005 2015 2014 2013 2014 1998 2015 2015 2018 2018 1998 2004 2015 2007 2018 2014 2014 2006 2002 2014 2004 2014 1998 2018

[57] 2014 2005 1997 1998 1999 2001 2001 1998 1997 2013 2018 2018 2004 2014 1997 2001 2013 2005 2015 2018 2014 2005 2018 2018 2014 2015 2005 2016

[85] 2013 2005 2018 2015 2016 2014 2014 2016 2014 2004 2015 2018 2004 2013 2016 2013 2018 2013 2015 2005 2018 2018 2018 2015 2006 2013 2018 2018

[113] 2018 2018 2018 2018 2018 2018 2018 2018 2018 2018 2018 2018 2018 2018 2018

> #цена

> price

[1] 435000 530000 432000 410000 245000 260000 438000 135000 519000 812517 449000 143000 480000 215000 427000 468000 207000 140000 390000 293000

[21] 430000 119000 409000 440000 500000 425000 395100 28000 85000 550000 135000 794538 235000 452124 350000 299000 390000 100000 498000 470000

[41] 741000 597000 40000 149990 498000 251000 741000 446800 440000 240000 245000 399000 165000 405000 130000 840832 384000 175000 60000 80000

[61] 33000 225000 50000 35000 50000 289999 597000 623000 226000 405000 70000 127000 450000 210000 430000 640000 450000 205000 803180 640000

[81] 450000 127000 230000 590000 470000 270000 703000 400000 574500 400000 460000 490000 380000 230000 515000 613000 230000 435000 520000 399000

[101] 727000 410000 440000 240000 570000 634000 603000 495000 260000 459000 605880 587000 770178 741000 741000 741000 741000 741000 741000 795831

[121] 790855 789459 738000 678000 703000 703000 784491

> #информация о характеристиках автомобиля

> info

[1] "122000км,1.6 AT (102л.с.),седан,передний,бензин"

"57000км,1.6 MT (102л.с.),седан,передний,бензин"

[3] "75623км,1.6 AT (102л.с.),седан,передний,бензин"

"111031км,1.6 MT (102л.с.),седан,передний,бензин"

[5] "174000км,1.5 MT (98л.с.),хетчбэк,передний,бензин"

"217990км,1.6 MT (107л.с.),седан,передний,бензин"

[7] "37959км,1.6 MT (102л.с.),седан,передний,бензин"

"180000км,1.8 MT (116л.с.),седан,передний,бензин"

[9] "62800км,1.6 MT (102л.с.),седан,передний,бензин"

"1.6 MT (102л.с.),седан,передний,бензин"

[11] "97840км,1.6 MT (102л.с.),седан,передний,бензин"

"180000км,1.6 MT (90л.с.),хетчбэк,передний,бензин"

[13] "35250км,1.6 AT (102л.с.),седан,передний,бензин"

"205000км,1.5 MT (98л.с.),седан,передний,бензин"

[15] "34500км,1.6 MT (102л.с.),седан,передний,бензин"

"48658км,1.6 MT (102л.с.),седан,передний,бензин"

[17] "107257км,1.5 MT (98л.с.),седан,передний,бензин"

"300000км,1.6 MT (99л.с.),седан,передний,бензин"

[19] "35000км,1.6 MT (102л.с.),седан,передний,бензин"

"130000км,1.6 MT (107л.с.),седан,передний,бензин"

> #ccылки на подробную информацию об автомобиле

> ref\_info

[1] "https://www.avito.ru/ulyanovsk/avtomobili/nissan\_almera\_2013\_1321065709"

[2] "https://www.avito.ru/kazan/avtomobili/nissan\_almera\_2016\_1611681345"

[3] "https://www.avito.ru/kazan/avtomobili/nissan\_almera\_2013\_1581885046"

[4] "https://www.avito.ru/kazan/avtomobili/nissan\_almera\_2013\_1592641850"

[5] "https://www.avito.ru/medvedevo/avtomobili/nissan\_almera\_2006\_1692980981"

[6] "https://www.avito.ru/cheboksary/avtomobili/nissan\_almera\_2007\_1273983685"

[7] "https://www.avito.ru/ulyanovsk/avtomobili/nissan\_almera\_2015\_1257270167"

[8] "https://www.avito.ru/kanash/avtomobili/nissan\_almera\_2006\_1335146710"

Некоторые автомобили являются битыми, сохраним информацию об этом в переменную is\_broken:

> #заполняем признак, является ли автомобиль битым, 1 - битый, 0 - нет

> is\_broken = ifelse(substr(info, 0, 5) == 'Битый', 1, 0)

> is\_broken

[1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0

[71] 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Теперь удалим из переменной info информацию о том, что автомобиль является битым:

> info = str\_replace(info, 'Битый,', '')

Характеристики автомобиля перечисляются через запятую, разобьем строку с помощью функции str\_split():

> split\_info <- str\_split(info, ',');

> split\_info

[[1]]

[1] "122000км" "1.6 AT (102л.с.)" "седан" "передний" "бензин"

[[2]]

[1] "57000км" "1.6 MT (102л.с.)" "седан" "передний" "бензин"

[[3]]

[1] "75623км" "1.6 AT (102л.с.)" "седан" "передний" "бензин"

Вытащим следующую информацию:

> mileage = numeric(); #пробег, км

> engine\_capacity = numeric(); #объем двигателя, л

> power = numeric(); #мощность л.с.

> transmission = character(); #трансмиссия (механика/автомат)

> body\_type = character(); #тип кузова

> drive\_unit = character(); #привод

> engine\_type = character(); #тип двигателя

Заметим, что если длина разбитой строки = 5, значит, в характеристиках есть информация о пробеге автомобиля, иначе – автомобиль новый, его пробег устанавливаем 0 км. С помощью регулярных выражений вытаскиваем нужную информацию:

> for(i in 1:length(split\_info))

+ {

+ str\_info = split\_info[[i]];

+ index = 1;

+

+ if(length(str\_info) == 5)

+ {

+ mileage = append(mileage, as.numeric(str\_extract(str\_info[1], "[:digit:]+")));

+ index = 2;

+ }

+ else

+ {

+ mileage = append(mileage, 0);

+ }

+

+ k = str\_extract(str\_info[index], "[(][:digit:]+");

+ k = as.numeric(str\_replace(k, '[(]', ''));

+ power = append(power, k);

+ y = str\_extract(str\_info[index], "[MA]?T");

+ transmission = append(transmission, y);

+ z = str\_extract(str\_info[index], "[:digit:]+[.][:digit:]+");

+ engine\_capacity = append(engine\_capacity, as.numeric(z))

+ body\_type = append(body\_type, str\_info[index+1])

+ drive\_unit = append(drive\_unit, str\_info[index+2])

+ engine\_type = append(engine\_type, str\_info[index+3])

+ }

> #пробег, км

> mileage

[1] 122000 57000 75623 111031 174000 217990 37959 180000 62800 0 97840 180000 35250 205000 34500 48658 107257 300000 35000 130000

[21] 89000 137000 82500 77000 85000 75000 56394 250000 350000 100000 265000 0 170000 97238 77145 142000 85000 150000 53300 13500

[41] 0 0 199999 224284 23500 224216 0 90000 56000 150000 190000 142500 49001 75101 189000 0 49388 156000 210111 273000

[61] 250000 233000 200000 200000 300000 155000 0 0 210000 98000 374000 200000 16000 250000 52000 200 42866 170000 0 21

[81] 73000 27000 190000 68400 49400 203000 0 20000 46250 80600 32000 36000 85000 170000 72000 0 250000 77000 22500 58000

[101] 0 63000 83000 113825 0 0 0 53672 127000 83000 0 0 0 0 0 0 0 0 0 0

[121] 0 0 0 0 0 0 0

> #мощность л.с.

> power

[1] 102 102 102 102 98 107 102 116 102 102 102 90 102 98 102 102 98 99 102 107 102 90 102 102 102 102 102 98 90 102 90 102 116 102 350

[36] 102 102 87 102 102 102 102 75 106 102 107 102 102 102 98 90 102 98 102 100 102 102 98 98 86 99 88 114 90 90 102 102 102 98 102

[71] 80 99 102 98 102 102 102 98 102 102 102 102 98 102 102 116 102 102 102 102 102 102 967 116 102 102 114 75 102 102 115 102 109 98 102

[106] 102 102 102 98 102 102 115 102 102 102 102 102 102 102 102 102 102 102 102 102 102 102

> #трансмиссия (механика/автомат)

> transmission

[1] "AT" "MT" "AT" "MT" "MT" "MT" "MT" "MT" "MT" "MT" "MT" "MT" "AT" "MT" "MT" "MT" "MT" "MT" "MT" "MT" "AT" "MT" "AT" "MT" "MT" "MT" "MT" "MT"

[29] "MT" "MT" "MT" "MT" "MT" "MT" "MT" "MT" "MT" "MT" "AT" "MT" "MT" "MT" "MT" "MT" "MT" "MT" "MT" "MT" "MT" "MT" "MT" "MT" "MT" "MT" "AT" "AT"

[57] "MT" "MT" "MT" "MT" "MT" "MT" "MT" "MT" "MT" "AT" "MT" "MT" "MT" "MT" "MT" "MT" "MT" "MT" "AT" "MT" "MT" "MT" "MT" "MT" "AT" "MT" "MT" "AT"

[85] "AT" "MT" "MT" "MT" "MT" "MT" "AT" "MT" "MT" "AT" "MT" "MT" "MT" "AT" "AT" "MT" "AT" "MT" "AT" "MT" "MT" "MT" "MT" "AT" "MT" "AT" "MT" "MT"

[113] "MT" "MT" "MT" "MT" "MT" "MT" "MT" "MT" "MT" "MT" "MT" "AT" "MT" "MT" "MT"

> #объем двигателя, л

> engine\_capacity

[1] 1.6 1.6 1.6 1.6 1.5 1.6 1.6 1.8 1.6 1.6 1.6 1.6 1.6 1.5 1.6 1.6 1.5 1.6 1.6 1.6 1.6 1.5 1.6 1.6 1.6 1.6 1.6 1.5 1.6 1.6 1.5 1.6 1.8 1.6 1.6

[36] 1.6 1.6 1.4 1.6 1.6 1.6 1.6 1.4 1.5 1.6 1.6 1.6 1.6 1.6 1.5 1.6 1.6 1.5 1.6 1.6 1.6 1.6 1.5 1.6 1.4 1.6 1.6 1.8 1.5 1.6 1.6 1.6 1.6 1.5 1.6

[71] 1.6 1.5 1.6 1.5 1.6 1.6 1.6 1.5 1.6 1.6 1.6 1.6 1.5 1.6 1.6 1.8 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.8 1.6 1.6 1.8 1.6 1.6 1.6 1.6 1.6 1.6 1.5 1.6

[106] 1.6 1.6 1.6 1.5 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6

> #тип кузова

> body\_type

[1] "седан" "седан" "седан" "седан" "хетчбэк" "седан" "седан" "седан" "седан" "седан" "седан" "хетчбэк" "седан" "седан"

[15] "седан" "седан" "седан" "седан" "седан" "седан" "седан" "хетчбэк" "седан" "седан" "седан" "седан" "седан" "седан"

[29] "седан" "седан" "хетчбэк" "седан" "хетчбэк" "седан" "седан" "седан" "седан" "хетчбэк" "седан" "седан" "седан" "седан"

[43] "хетчбэк" "седан" "седан" "седан" "седан" "седан" "седан" "седан" "седан" "седан" "седан" "седан" "хетчбэк" "седан"

[57] "седан" "седан" "седан" "седан" "седан" "седан" "хетчбэк" "седан" "хетчбэк" "седан" "седан" "седан" "седан" "седан"

[71] "седан" "седан" "седан" "седан" "седан" "седан" "седан" "седан" "седан" "седан" "седан" "седан" "седан" "седан"

[85] "седан" "хетчбэк" "седан" "седан" "седан" "седан" "седан" "седан" "седан" "седан" "седан" "седан" "седан" "седан"

[99] "седан" "седан" "седан" "седан" "седан" "седан" "седан" "седан" "седан" "седан" "седан" "седан" "седан" "седан"

[113] "седан" "седан" "седан" "седан" "седан" "седан" "седан" "седан" "седан" "седан" "седан" "седан" "седан" "седан"

[127] "седан"

> #привод

> drive\_unit

[1] "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний"

[13] "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний"

[25] "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний"

[37] "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний"

[49] "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний"

[61] "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний"

[73] "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний"

[85] "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний"

[97] "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний"

[109] "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний" "передний"

[121] "передний" "передний" "передний" "передний" "передний" "передний" "передний"

> #тип двигателя

> engine\_type

[1] "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин"

[16] "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин"

[31] "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин"

[46] "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин"

[61] "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин"

[76] "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин"

[91] "бензин" "бензин" "газ" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин"

[106] "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин"

[121] "бензин" "бензин" "бензин" "бензин" "бензин" "бензин" "бензин"

Запишем в переменную petgas информацию о типе двигателя:

> petgas = numeric(); #тип двигателя - бензин(0) /газ (1)

> for(i in 1:length(engine\_type))

+ {

+ if(engine\_type[i] == 'бензин')

+ {

+ petgas[i] = 0;

+ }

+ else

+ {

+ petgas[i] = 1;

+ }

+

+ }

> petgas

[1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

[71] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Запишем в переменную hs информацию о типе кузова:

> hs = numeric(); #тип кузова - хетчбэк(0) / седан(1)

> for(i in 1:length(body\_type))

+ {

+ if(body\_type[i] == 'хетчбэк')

+ {

+ hs[i] = 0;

+ }

+ else

+ {

+ hs[i] = 1;

+ }

+

+ }

> hs

[1] 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1

[71] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

> # вытащим информацию о цвете автомобиля

> for(i in 1:length(ref\_info))

+ {

+   page <- read\_html(ref\_info[i])

+   results <- page %>% html\_nodes(".item-params-list-item") %>% html\_text() %>% str\_trim()

+   for(j in 1:length(results))

+   {

+     if(substr(results[j], 0, 4) == 'Цвет')

+     {

+       col <- substr(results[j], 6, nchar(results[j])) %>% str\_trim()

+       color = append(color, col)

+     }

+   }

+ }

> #цвет

> color

[1] "бежевый" "бежевый" "коричневый" "синий" "серый" "синий" "чёрный" "серебряный" "чёрный" "белый"

[11] "серебряный" "зелёный" "белый" "чёрный" "чёрный" "синий" "голубой" "зелёный" "серый" "серебряный"

[21] "синий" "синий" "серебряный" "серый" "чёрный" "белый" "синий" "серый" "серебряный" "белый"

[31] "синий" "бежевый" "серебряный" "чёрный" "чёрный" "синий" "белый" "красный" "серебряный" "белый"

[41] "бежевый" "серый" "зелёный" "синий" "синий" "серебряный" "чёрный" "чёрный" "серебряный" "бежевый"

[51] "серебряный" "чёрный" "золотой" "белый" "белый" "чёрный" "серебряный" "серый" "красный" "зелёный"

[61] "зелёный" "серебряный" "серебряный" "зелёный" "чёрный" "синий" "чёрный" "белый" "золотой" "синий"

[71] "белый" "серебряный" "серебряный" "чёрный" "бежевый" "белый" "белый" "серебряный" "белый" "белый"

[81] "красный" "бежевый" "серый" "белый" "чёрный" "серебряный" "серебряный" "серебряный" "синий" "белый"

[91] "чёрный" "чёрный" "чёрный" "серый" "красный" "синий" "серебряный" "золотой" "серый" "белый"

[101] "чёрный" "бежевый" "белый" "чёрный" "белый" "чёрный" "синий" "белый" "серебряный" "красный"

[111] "бежевый" "серебряный" "чёрный" "серебряный" "чёрный" "чёрный" "чёрный" "белый" "серебряный" "чёрный"

[121] "белый" "белый" "бежевый" "чёрный" "бежевый" "чёрный" "белый"

Запишем в переменную col информацию о цвете автомобиля:

> col = numeric(); #цвет автомобиля серебряный, черный или белый (0) /

> for(i in 1:length(color))

+ {

+ if(color[i] == 'черный' || color[i] == 'белый' || color[i] == 'серебряный')

+ {

+ col[i] = 0;

+ }

+ else

+ {

+ col[i] = 1;

+ }

+ }

> col

[1] 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 1 0 1 0 0 1 1 1 1 1 0 1 1 0 1 0 1 1 0 0 1 0 1 1 1 1 0 0 1 1 1 1 0 1 1

[71] 0 0 0 1 1 0 0 0 0 0 1 1 1 0 1 0 0 0 1 0 1 1 1 1 1 1 0 1 1 0 1 1 0 1 0 1 1 0 0 1 1 0 1 0 1 1 1 0 0 1 0 0 1 1 1 1 0

Построим модель для прогнозирования. Входные параметры для модели прогноза:

* Год выпуска;
* Признак битый / нет;
* Пробег, км;
* Мощность, л.с.;
* Объем двигателя, л;
* Признак тип кузова: хетчбэк /седан;
* Признак тип двигателя: бензин/газ;
* Признак: цвет является стандартным (черный, белый, серебряный) /нет.

Выходной параметр – цена.

Для удобства разделим пробег и цену на 1000:

> mileage = mileage / 1000;

> price = price / 1000;

> # сохраним преобразованные поля как data.frame

> parse\_data = data.frame("year" = year, "is\_broken" = is\_broken, "mileage" = mileage, "power" = power,

+ "engine\_capacity" = engine\_capacity, "hs" = hs, "petgas" = petgas,

+ "col" = col, "price" = price);

> parse\_data

year is\_broken mileage power engine\_capacity hs petgas col price

1 2013 0 122.000 102 1.6 1 0 1 435.000

2 2016 0 57.000 102 1.6 1 0 1 530.000

3 2013 0 75.623 102 1.6 1 0 1 432.000

4 2013 0 111.031 102 1.6 1 0 1 410.000

5 2006 0 174.000 98 1.5 0 0 1 245.000

6 2007 0 217.990 107 1.6 1 0 1 260.000

7 2015 0 37.959 102 1.6 1 0 1 438.000

8 2006 0 180.000 116 1.8 1 0 0 135.000

9 2016 0 62.800 102 1.6 1 0 1 519.000

10 2018 0 0.000 102 1.6 1 0 0 812.517

11 2015 0 97.840 102 1.6 1 0 0 449.000

12 2001 0 180.000 90 1.6 0 0 1 143.000

13 2013 0 35.250 102 1.6 1 0 0 480.000

14 2005 0 205.000 98 1.5 1 0 1 215.000

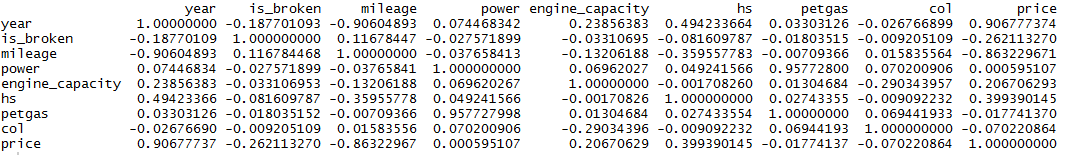
15 2013 0 34.500 102 1.6 1 0 1 427.000

Вычислим попарные коэффициенты корреляции между предикторами:

> # вычислим попарные коэффициенты корреляции между предикторами

> r = cor(parse\_data)

> r



Как видим, имеется сильная отрицательная корреляция между пробегом и годом выпуска (-0.906), сильная положительная корреляция между мощностью и признаком тип двигателя бензин/газ (0.958).

> # построим регрессионную модель

> mymodel = lm(price ~ year + is\_broken + mileage + power + engine\_capacity + hs + petgas + col, data = parse\_data)

> mymodel

Call:

lm(formula = price ~ year + is\_broken + mileage + power + engine\_capacity +

hs + petgas + col, data = parse\_data)

Residuals:

Min 1Q Median 3Q Max

-287.31 -57.72 -16.12 52.58 216.75

> summary(mymodel)

Call:

lm(formula = price ~ year + is\_broken + mileage + power + engine\_capacity +

hs + petgas + col, data = parse\_data)

Residuals:

Min 1Q Median 3Q Max

-287.31 -57.72 -16.12 52.58 216.75

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -4.595e+04 6.667e+03 -6.892 2.88e-10 \*\*\*

year 2.311e+01 3.348e+00 6.904 2.71e-10 \*\*\*

is\_broken -1.246e+02 4.119e+01 -3.025 0.00305 \*\*

mileage -5.863e-01 2.206e-01 -2.658 0.00895 \*\*

power -6.751e-01 3.495e-01 -1.932 0.05578 .

engine\_capacity 2.639e+01 1.381e+02 0.191 0.84879

hs -2.538e+01 3.359e+01 -0.756 0.45138

petgas 4.821e+02 3.136e+02 1.538 0.12683

col -1.964e+01 1.693e+01 -1.161 0.24812

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 87.74 on 118 degrees of freedom

Multiple R-squared: 0.8529, Adjusted R-squared: 0.8429

F-statistic: 85.53 on 8 and 118 DF, p-value: < 2.2e-16

Заметим, что коэффициент детерминации = 0.8529, «исправленный» коэффициент детерминации = 0.8429.

Построим регрессионную модель, где предикторы год, признак битый / нет, пробег.

> # построим регрессионную модель

> mymodel = lm(price ~ year + is\_broken + mileage, data = parse\_data)

> mymodel

Call:

lm(formula = price ~ year + is\_broken + mileage, data = parse\_data)

Coefficients:

(Intercept) year is\_broken mileage

-4.275e+04 2.149e+01 -1.246e+02 -6.509e-01

> summary(mymodel)

Call:

lm(formula = price ~ year + is\_broken + mileage, data = parse\_data)

Residuals:

Min 1Q Median 3Q Max

-293.89 -58.64 -17.50 60.68 213.25

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -4.275e+04 5.773e+03 -7.404 1.81e-11 \*\*\*

year 2.149e+01 2.861e+00 7.512 1.03e-11 \*\*\*

is\_broken -1.246e+02 4.154e+01 -3.001 0.00326 \*\*

mileage -6.509e-01 2.086e-01 -3.120 0.00225 \*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 88.67 on 123 degrees of freedom

Multiple R-squared: 0.8434, Adjusted R-squared: 0.8396

F-statistic: 220.8 on 3 and 123 DF, p-value: < 2.2e-16

Заметим, что коэффициент детерминации уменьшился до 0.8434, «исправленный» коэффициент детерминации уменьшился до 0.8396. Поэтому воспользуемся первой моделью.

> # спрогнозируем цену автомобилей по нашей модели

> predictPrice = predict(mymodel, predict\_data)

> predictPrice = as.data.frame(predictPrice)

> predictPrice

predictPrice

1 436.992833

2 544.441118

3 464.184653

4 443.424191

5 270.160274

6 238.661683

7 532.492792

8 256.669746

9 541.040455

10 643.730669

> # истинная цена автомобилей

> truePrice = parse\_data$price

> truePrice

[1] 435.000 530.000 432.000 410.000 245.000 260.000 438.000 135.000 519.000 812.517 449.000 143.000 480.000 215.000 427.000 468.000 207.000 140.000 390.000 293.000

[21] 430.000 119.000 409.000 440.000 500.000 425.000 395.100 28.000 85.000 550.000 135.000 794.538 235.000 452.124 350.000 299.000 390.000 100.000 498.000 470.000

[41] 741.000 597.000 40.000 149.990 498.000 251.000 741.000 446.800 440.000 240.000 245.000 399.000 165.000 405.000 130.000 840.832 384.000 175.000 60.000 80.000

[61] 33.000 225.000 50.000 35.000 50.000 289.999 597.000 623.000 226.000 405.000 70.000 127.000 450.000 210.000 430.000 640.000 450.000 205.000 803.180 640.000

[81] 450.000 127.000 230.000 590.000 470.000 270.000 703.000 400.000 574.500 400.000 460.000 490.000 380.000 230.000 515.000 613.000 230.000 435.000 520.000 399.000

[101] 727.000 410.000 440.000 240.000 570.000 634.000 603.000 495.000 260.000 459.000 605.880 587.000 770.178 741.000 741.000 741.000 741.000 741.000 741.000 795.831

[121] 790.855 789.459 738.000 678.000 703.000 703.000 784.491

Вычислим ошибку прогноза в процентах. Найдем, на сколько процентов отличается разница между истинной и предсказанной ценами автомобилей.

> # ошибка прогноза

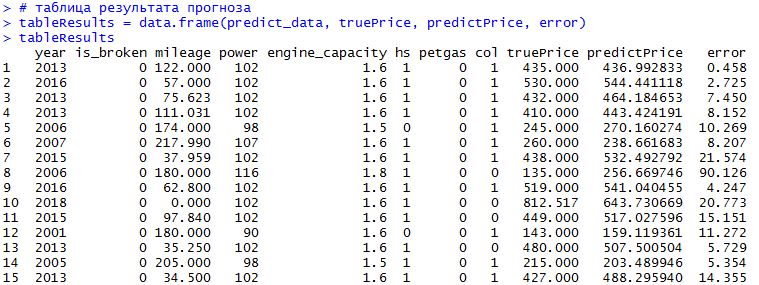
> error = abs(truePrice - predictPrice)\*100/truePrice;

> error= round(error, 3)

> colnames(error) = "error"

> max(error)

[1] 615.064



Выберем те автомобили, для которых ошибка прогноза составила больше 50%:

> for(i in 1:nrow(error))

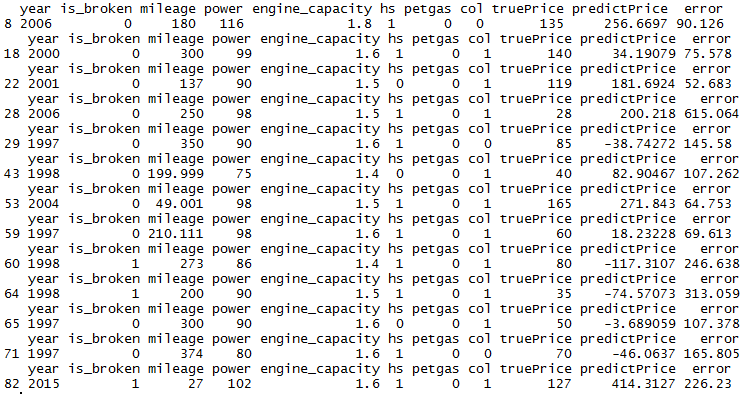
+ {

+ if(error[i,] > 50)

+ print(tableResults[i,]);

+ }

Получим 13 автомобилей:



Самая большая ошибка прогноза (615.064%) у автомобиля 28. Автомобиль 28 имеет признак не битый, однако в подробной информации на сайте указано о том, что у него разбита задняя часть. Автомобили под номерами 60, 64, 82 являются битыми. Остальные автомобили хоть и имеют признак не битый, на сайте в подробной информации владельцы указали, что машины имеют повреждения.

Исключим из выборки те автомобили, ошибка прогноза для которых больше 100 %.

Получим номера этих автомобилей:

> index = numeric();

> for(i in 1:nrow(error))

+ {

+ if(error[i,] > 100)

+ index = append(index, i)

+ }

> index

[1] 28 29 43 60 64 65 71 82

> #исключаем их из выборки

> new\_data = parse\_data[-c(index),]

Построим регрессионную модель для новых данных:

> # построим регрессионную модель

> mymodel = lm(price ~ year + is\_broken + mileage + power + engine\_capacity + hs + petgas + col, data = new\_data)

> mymodel

Call:

lm(formula = price ~ year + is\_broken + mileage + power + engine\_capacity +

hs + petgas + col, data = new\_data)

Coefficients:

(Intercept) year is\_broken mileage power engine\_capacity hs petgas col

-4.781e+04 2.401e+01 -1.368e+02 -7.057e-01 -6.000e-01 7.739e+01 -4.987e+01 4.146e+02 -1.007e+01

> summary(mymodel)

Call:

lm(formula = price ~ year + is\_broken + mileage + power + engine\_capacity +

hs + petgas + col, data = new\_data)

Residuals:

Min 1Q Median 3Q Max

-135.54 -58.41 -10.42 47.31 200.47

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -4.781e+04 6.252e+03 -7.648 8.29e-12 \*\*\*

year 2.401e+01 3.140e+00 7.647 8.33e-12 \*\*\*

is\_broken -1.368e+02 5.972e+01 -2.290 0.02391 \*

mileage -7.057e-01 2.183e-01 -3.233 0.00162 \*\*

power -6.000e-01 3.168e-01 -1.894 0.06088 .

engine\_capacity 7.739e+01 1.359e+02 0.569 0.57021

hs -4.987e+01 3.350e+01 -1.489 0.13944

petgas 4.146e+02 2.840e+02 1.460 0.14720

col -1.007e+01 1.566e+01 -0.643 0.52150

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 79.05 on 110 degrees of freedom

Multiple R-squared: 0.8643, Adjusted R-squared: 0.8544

F-statistic: 87.56 on 8 and 110 DF, p-value: < 2.2e-16

Заметим, что коэффициент детерминации увеличился до 0.8643, «исправленный» коэффициент детерминации увеличился до 0.8544.

> # спрогнозируем цену автомобилей по нашей модели

> predictPrice = predict(mymodel, predict\_data)

> predictPrice = as.data.frame(predictPrice)

> predictPrice

predictPrice

1 434.218108

2 552.119402

3 466.947951

4 441.959308

5 273.985024

6 219.417670

7 541.547770

8 242.368539

9 548.026142

10 650.436276

11 509.358666

12 162.241993

13 505.511540

14 178.226727

15 495.969865

> # истинная цена автомобилей

> truePrice = new\_data$price

> truePrice

[1] 435.000 530.000 432.000 410.000 245.000 260.000 438.000 135.000 519.000 812.517 449.000 143.000 480.000 215.000 427.000 468.000 207.000 140.000 390.000 293.000

[21] 430.000 119.000 409.000 440.000 500.000 425.000 395.100 550.000 135.000 794.538 235.000 452.124 350.000 299.000 390.000 100.000 498.000 470.000 741.000 597.000

[41] 149.990 498.000 251.000 741.000 446.800 440.000 240.000 245.000 399.000 165.000 405.000 130.000 840.832 384.000 175.000 60.000 33.000 225.000 50.000 289.999

[61] 597.000 623.000 226.000 405.000 127.000 450.000 210.000 430.000 640.000 450.000 205.000 803.180 640.000 450.000 230.000 590.000 470.000 270.000 703.000 400.000

[81] 574.500 400.000 460.000 490.000 380.000 230.000 515.000 613.000 230.000 435.000 520.000 399.000 727.000 410.000 440.000 240.000 570.000 634.000 603.000 495.000

[101] 260.000 459.000 605.880 587.000 770.178 741.000 741.000 741.000 741.000 741.000 741.000 795.831 790.855 789.459 738.000 678.000 703.000 703.000 784.491

> # ошибка прогноза

> error = abs(truePrice - predictPrice)\*100/truePrice;

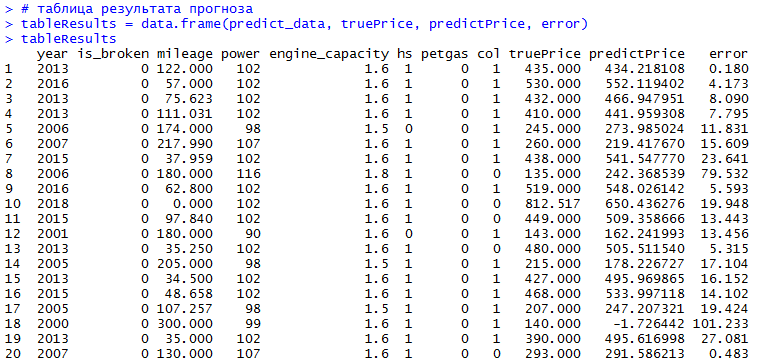
> error= round(error, 3)

> colnames(error) = "error"

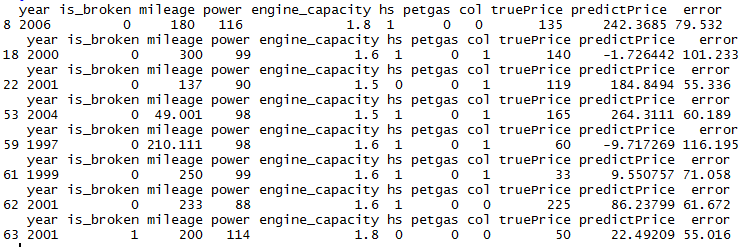
> max(error)

[1] 116.195

Как видим, максимальная ошибка прогноза уменьшилась в 6 раз.



Выберем те автомобили, для которых ошибка прогноза составила больше 50%:



Все автомобили являются битыми.

Таким образом, для улучшения модели, необходимо из выборки убрать те автомобили, для которых ошибка прогноза составляет больше 100 %.

**Код программы:**

# назначим рабочий директорий

setwd("C:/Users/TatyanaAnisimova/Documents/Rprojects")

library(rvest)

library(stringr)

main\_url = 'https://www.avito.ru'

nissan\_url= 'https://www.avito.ru/kazan/avtomobili/nissan/almera?radius=200'

short\_url = str\_replace(nissan\_url, 'radius=200', '');

pages <- read\_html(nissan\_url)

pages

# функция для получения номера последней страницы

get\_last\_page\_numb <- function(html){

pages\_data <- html %>%

html\_nodes('.pagination-page') %>%

html\_text()

pages\_data[(length(pages\_data)-1)] %>%

unname() %>%

as.numeric()

}

# получим номер последней страницы

(last\_page\_numb <- get\_last\_page\_numb(pages))

# получим список страниц

list\_of\_pages = character();

list\_of\_pages = append(list\_of\_pages, nissan\_url)

list\_of\_pages = append(list\_of\_pages, str\_c(short\_url, paste0('p=', 2:last\_page\_numb, '&radius=200')));

list\_of\_pages

year = numeric(); #год выпуска

price = numeric(); #цена

info = character(); #характеристики

ref\_info = character(); #ссылки

is\_broken = numeric(); #признак, является ли автомобиль битым, (1 - является, 0 - нет)

color = character(); #цвет автомобиля

for(i in 1:last\_page\_numb)

{

page <- read\_html(list\_of\_pages[i]);

results <- page %>% html\_nodes(".item\_hide-elements")

Sys.sleep(1)

#вытаскиваем год выпуска

tmp <- results %>% html\_nodes(".item-description-title")%>% html\_text()%>%str\_trim()

year\_tmp = as.numeric(str\_extract(tmp, "[:digit:]+"));

year = append(year, year\_tmp);

Sys.sleep(1)

#вытаскиваем цену автомобиля

price\_tmp <- results %>% html\_nodes(".about\_bold-price") %>% html\_nodes(".price") %>% html\_text()

price\_tmp = as.numeric(str\_replace\_all(price\_tmp, "[ \n\r\t\\\u20bd]+", ""))

price = append(price, price\_tmp);

Sys.sleep(1)

#вытаскиваем информацию о характеристиках автомобиля

info\_tmp <- results %>% html\_nodes(".about\_bold-price")%>% html\_nodes(".specific-params\_block")%>% html\_text()%>%str\_trim()

info\_tmp = str\_replace\_all(info\_tmp, "[ ]+", "");

info = append(info, info\_tmp);

Sys.sleep(1)

#вытаскиваем ccылки на подробную информацию об автомобиле

ref <- results %>% html\_nodes(".item-description-title-link") %>% html\_attr("href")

ref <- paste0(main\_url, ref)

ref\_info = append(ref\_info, ref)

}

#год выпуска

year

#цена

price

#информация о характеристиках автомобиля

info = str\_replace\_all(info, "\"", "")

info

#ccылки на подробную информацию об автомобиле

ref\_info = str\_replace\_all(ref\_info, "\"", "")

ref\_info

#заполняем признак, является ли автомобиль битым, 1 - битый, 0 - нет

is\_broken = ifelse(substr(info, 0, 5) == 'Битый', 1, 0)

is\_broken

info = str\_replace(info, 'Битый,', '')

info

split\_info <- str\_split(info, ',');

split\_info

mileage = numeric(); #пробег, км

engine\_capacity = numeric(); #объем двигателя, л

power = numeric(); #мощность л.с.

transmission = character(); #трансмиссия (механика/автомат)

body\_type = character(); #тип кузова

drive\_unit = character(); #привод

engine\_type = character(); #тип двигателя

for(i in 1:length(split\_info))

{

str\_info = split\_info[[i]];

index = 1;

if(length(str\_info) == 5)

{

mileage = append(mileage, as.numeric(str\_extract(str\_info[1], "[:digit:]+")));

index = 2;

}

else

{

mileage = append(mileage, 0);

}

k = str\_extract(str\_info[index], "[(][:digit:]+");

k = as.numeric(str\_replace(k, '[(]', ''));

power = append(power, k);

y = str\_extract(str\_info[index], "[MA]?T");

transmission = append(transmission, y);

z = str\_extract(str\_info[index], "[:digit:]+[.][:digit:]+");

engine\_capacity = append(engine\_capacity, as.numeric(z))

body\_type = append(body\_type, str\_info[index+1])

drive\_unit = append(drive\_unit, str\_info[index+2])

engine\_type = append(engine\_type, str\_info[index+3])

}

#пробег, км

mileage

#мощность л.с.

power

#трансмиссия (механика/автомат)

transmission

#объем двигателя, л

engine\_capacity

#тип кузова

body\_type

#привод

drive\_unit

#тип двигателя

engine\_type

petgas = numeric(); #тип двигателя - бензин(0) /газ (1)

for(i in 1:length(engine\_type))

{

if(engine\_type[i] == 'бензин')

{

petgas[i] = 0;

}

else

{

petgas[i] = 1;

}

}

petgas

hs = numeric(); #тип кузова - хетчбэк(0) / седан(1)

for(i in 1:length(body\_type))

{

if(body\_type[i] == 'хетчбэк')

{

hs[i] = 0;

}

else

{

hs[i] = 1;

}

}

hs

# вытащим информацию о цвете автомобиля

for(i in 1:length(ref\_info))

{

Sys.sleep(5)

page <- read\_html(ref\_info[i])

Sys.sleep(5)

results <- page %>% html\_nodes(".item-params-list-item") %>% html\_text() %>% str\_trim()

Sys.sleep(5)

for(j in 1:length(results))

{

if(substr(results[j], 0, 4) == 'Цвет')

{

col <- substr(results[j], 6, nchar(results[j])) %>% str\_trim()

color = append(color, col)

}

}

}

#цвет

color

col = numeric(); #цвет автомобиля серебряный, черный или белый (0) /

#цвет автомобиля любой другой (1)

for(i in 1:length(color))

{

if(color[i] == 'черный' || color[i] == 'белый' || color[i] == 'серебряный')

{

print(color[i])

col[i] = 0;

}

else

{

col[i] = 1;

}

}

col

mileage = mileage / 1000;

price = price / 1000;

# сохраним преобразованные поля как data.frame

parse\_data = data.frame("year" = year, "is\_broken" = is\_broken, "mileage" = mileage, "power" = power,

"engine\_capacity" = engine\_capacity, "hs" = hs, "petgas" = petgas,

"col" = col, "price" = price);

parse\_data

# информация о данных

summary(parse\_data)

# вычислим попарные коэффициенты корреляции между предикторами

r = cor(parse\_data)

r

# построим регрессионную модель

mymodel = lm(price ~ year + is\_broken + mileage + power + engine\_capacity + hs + petgas + col, data = parse\_data)

mymodel

summary(mymodel)

# построим регрессионную модель

mymodel = lm(price ~ year + is\_broken + mileage, data = parse\_data)

mymodel

summary(mymodel)

# данные для прогноза

predict\_data = parse\_data

predict\_data = predict\_data[-9]

predict\_data

# спрогнозируем цену автомобилей по нашей модели

predictPrice = predict(mymodel, predict\_data)

predictPrice = as.data.frame(predictPrice)

predictPrice

# истинная цена автомобилей

truePrice = parse\_data$price

truePrice

# ошибка прогноза

error = abs(truePrice - predictPrice)\*100/truePrice;

error= round(error, 3)

colnames(error) = "error"

max(error)

# таблица результата прогноза

tableResults = data.frame(predict\_data, truePrice, predictPrice, error)

tableResults

for(i in 1:nrow(error))

{

if(error[i,] > 50)

print(tableResults[i,]);

}

index = numeric();

for(i in 1:nrow(error))

{

if(error[i,] > 100)

index = append(index, i)

}

index

#исключаем их из выборки

new\_data = parse\_data[-c(index),]

new\_data

# построим регрессионную модель

mymodel = lm(price ~ year + is\_broken + mileage + power + engine\_capacity + hs + petgas + col, data = new\_data)

mymodel

summary(mymodel)

# данные для прогноза

predict\_data = new\_data

predict\_data = predict\_data[-9]

predict\_data

# спрогнозируем цену автомобилей по нашей модели

predictPrice = predict(mymodel, predict\_data)

predictPrice = as.data.frame(predictPrice)

predictPrice

# истинная цена автомобилей

truePrice = new\_data$price

truePrice

# ошибка прогноза

error = abs(truePrice - predictPrice)\*100/truePrice;

error= round(error, 3)

colnames(error) = "error"

max(error)

# таблица результата прогноза

tableResults = data.frame(predict\_data, truePrice, predictPrice, error)

tableResults

for(i in 1:nrow(error))

{

if(error[i,] > 50)

print(tableResults[i,]);

}