Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Уральский федеральный университет имени первого Президента России Б.Н. Ельцина» (УрФУ)

Институт радиоэлектроники и информационных технологий – РтФ

Школа бакалавриата

Оценка

Дата защиты

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к проекту по модулю «Методы анализа Big Data» по теме «Зависимость количества заявок от активности аудитории в социальных сетях»

Подпись Дата

Преподаватель С.Г. Мирвода

Студенты Д.С. Евдокимова

Р.В. Кочеров

Группа РИ-450005

**Содержание**

[Введение 3](#_Toc535499155)

[1 Описание набора данных 4](#_Toc535499156)

[2 Обработка данных 6](#_Toc535499157)

[3 Построение моделей 7](#_Toc535499158)

[4 Разработка сервиса 13](#_Toc535499159)

[Заключение 15](#_Toc535499160)

[Библиографический список 16](#_Toc535499161)

[ПРИЛОЖЕНИЕ А 17](#_Toc535499162)

[ПРИЛОЖЕНИЕ B 18](#_Toc535499163)

# Введение

С развитием современных технологий появились новые возможности для анализа огромного количества данных. Big data обрабатываются для того, чтобы человек мог получить конкретные необходимые ему результаты для их дальнейшего эффективного применения.

В ходе работы над данным проектом в качестве предмета исследования была выбрана сфера продвижения в социальных сетях. В современном мире пользователи все больше времени проводят в социальных сетях, и все компании стремятся представить там свои товары и услуги, открывая тематические блоги и профили компании, а также принимая заявки. Так как все компании стремятся увеличить количество заявок для получения максимальной прибыли, для исследования было решено проверить зависимость количества поступающих заявок от откликов аудитории в социальных сетях на странице профиля компании.

Для реализации цели проекта были поставлены следующие задачи:

1. Выбор и обработка соответствующего набора данных.
2. Анализ данных.
3. Фильтрация набора данных.
4. Построение моделей данных.
5. Разработка приложения для удобной работы с созданной моделью.

# Описание набора данных

Одной из самых популярных социальных сетей в России на данным момент является «Вконтакте», большинство компаний активно ведут профили группы и покупают платную рекламу в данной социальной сети. Данная социальная сеть также примечательна тем, что имеет собственные сервисы подсчета статистики и предоставляет возможность выгрузить данные сразу в csv-файл.

Для нашего исследования было взято два набора данных. В файл *vk\_stat.csv* была выгружена статистика группы Вконтакте некоторой организации за период с 20 сентября 2018 по 14 января 2019, то есть почти целый квартал. Вво второй файл *applications.csv* были выгружены данные о заявках, поступающих через виджет группы, за тот же период.

Файл *vk\_stat.csv* содержит в себе следующие колонки:

* "Date",
* "Criterion",
* "Parameter 1",
* "Parameter 2",
* "Value".

Среди этих данных нас интересовали значения *feedback* в колонке *Criterion,* параметр 1 которых соответственно указывал *Likes, Comments* и *Shares* (отметки нравится, комментарии и репосты). Наиболее показательными являются лайки, поэтому при постройке моделей был использован именно этот параметр. Данных о комментариях и репостах оказалось недостаточно для построения релевантной модели.

Файл *applications.csv* содержит в себе следующие колонки:

* "Date",
* "Number",
* "Status",
* "Comment",
* "id",
* "User.first.name",
* "User.last.name",
* "Birthday",
* "Sex",
* "City",
* "Country",
* "First.name",
* "Last.name",
* "Phone",
* "Mail",
* "User",
* "Link".

# Обработка данных

Сгенерированный *CSV* файл был подгружен в *R* скрипт, после чего были исправлены колонки таблицы, которые были неправильно прочитаны языком *R*. Даты были переведены в правильный вид. Для удобства обработанная таблица была отдельно сохранена в формате *Rda*, далее используемом для анализа. Данные из файла *vk\_stat.csv* были также выгружены в отдельные таблицы для удобства дальнейшей обработки.

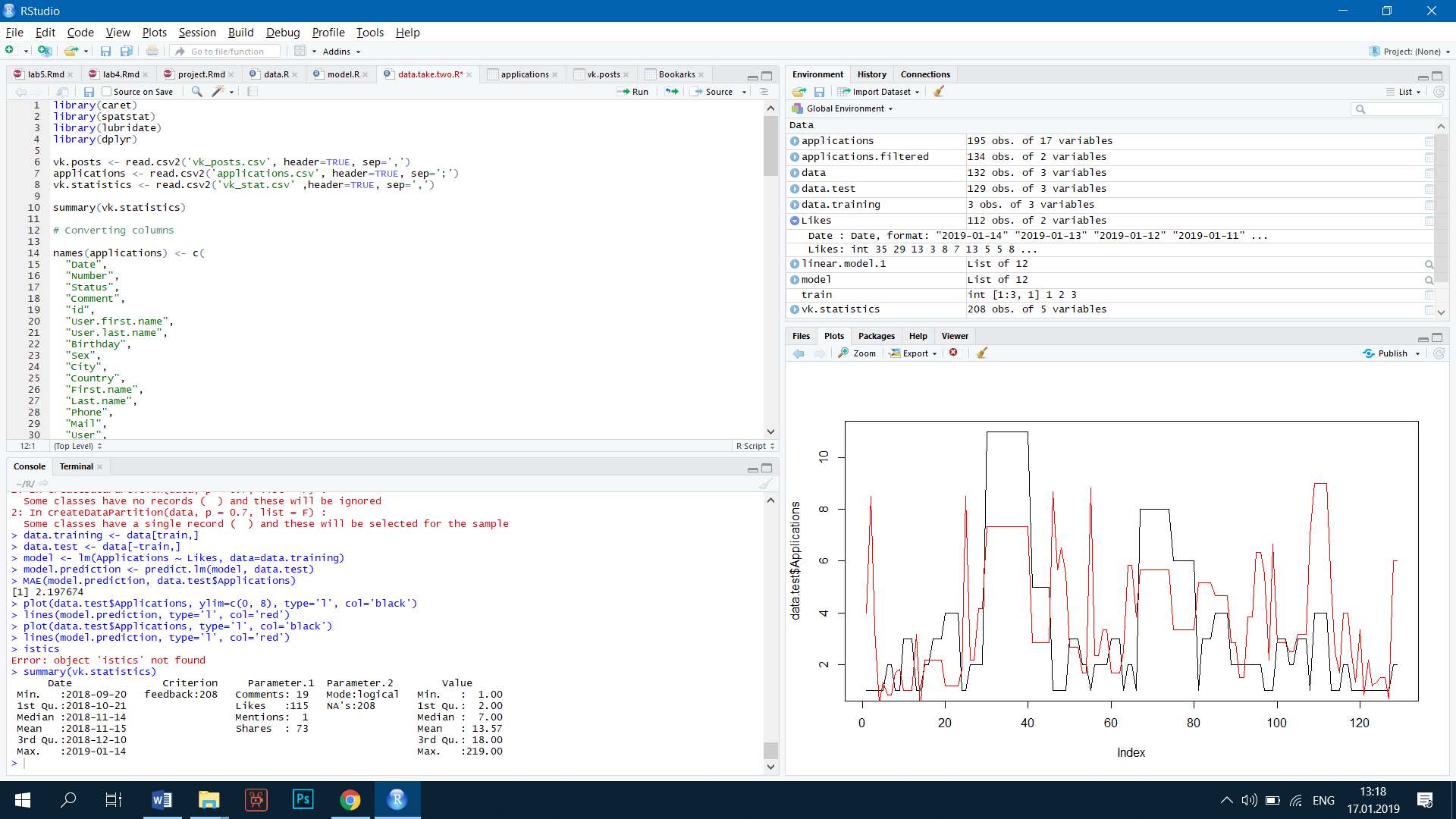


Рисунок 1 – Результат выполнения функции summary() на наборе данных

# Построение моделей

Для построения моделей была выбрана модель *lm* *(англ.* *Linear model function, линейная регрессионная модель*).

Изначально были построены графики количества заявок, лайков, комментариев и репостов по дате, которые представлены на рисунках 2-3.

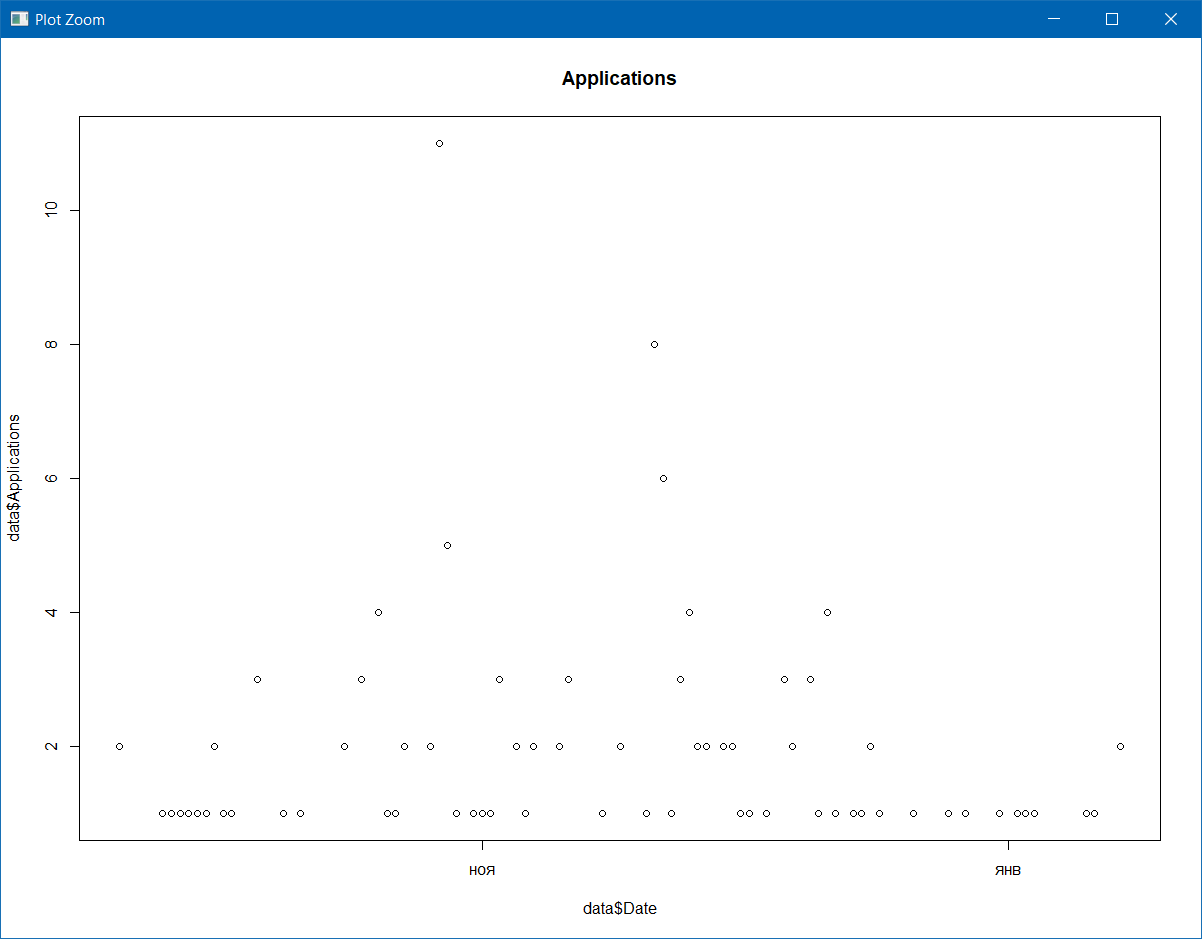


Рисунок 2 – Графики количества заявок по дням

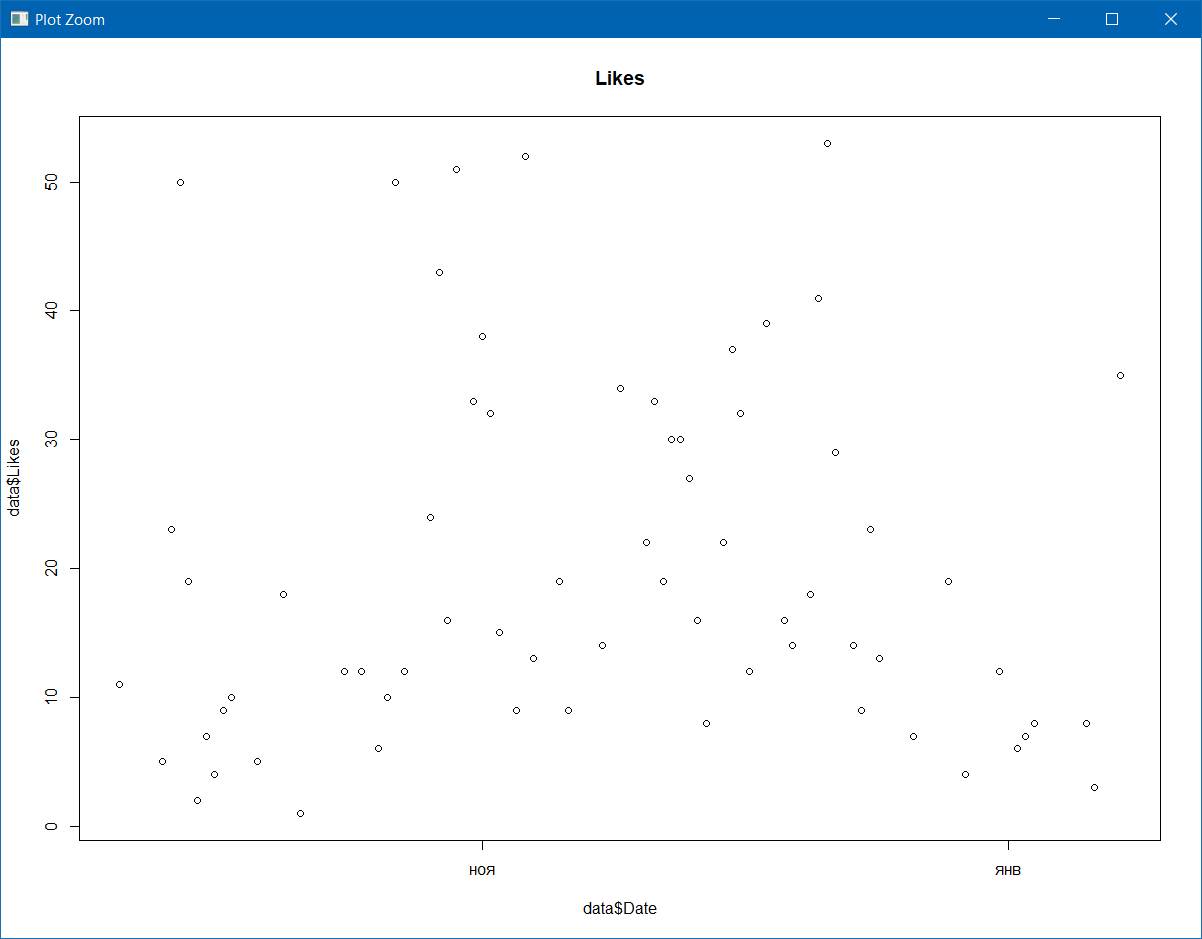


Рисунок 3 – Графики количества лайков по дням

Далее оценка был проведена на основе графиков boxplot(), приведенных на рисунке 4.

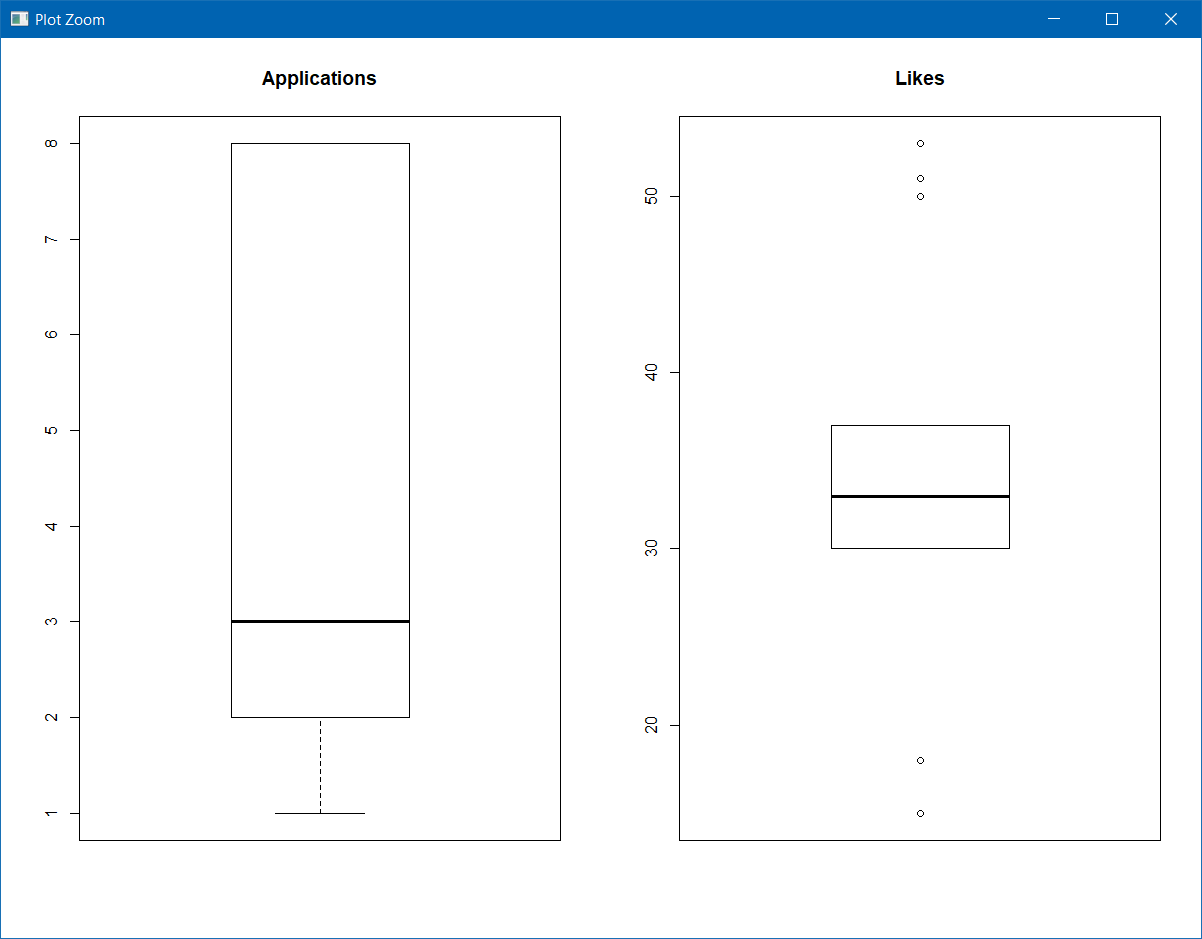


Рисунок 4 – Графики количества заявок и лайков boxplot()

Также для оценки был построен график рассеяния, представленный на рисунке 5.

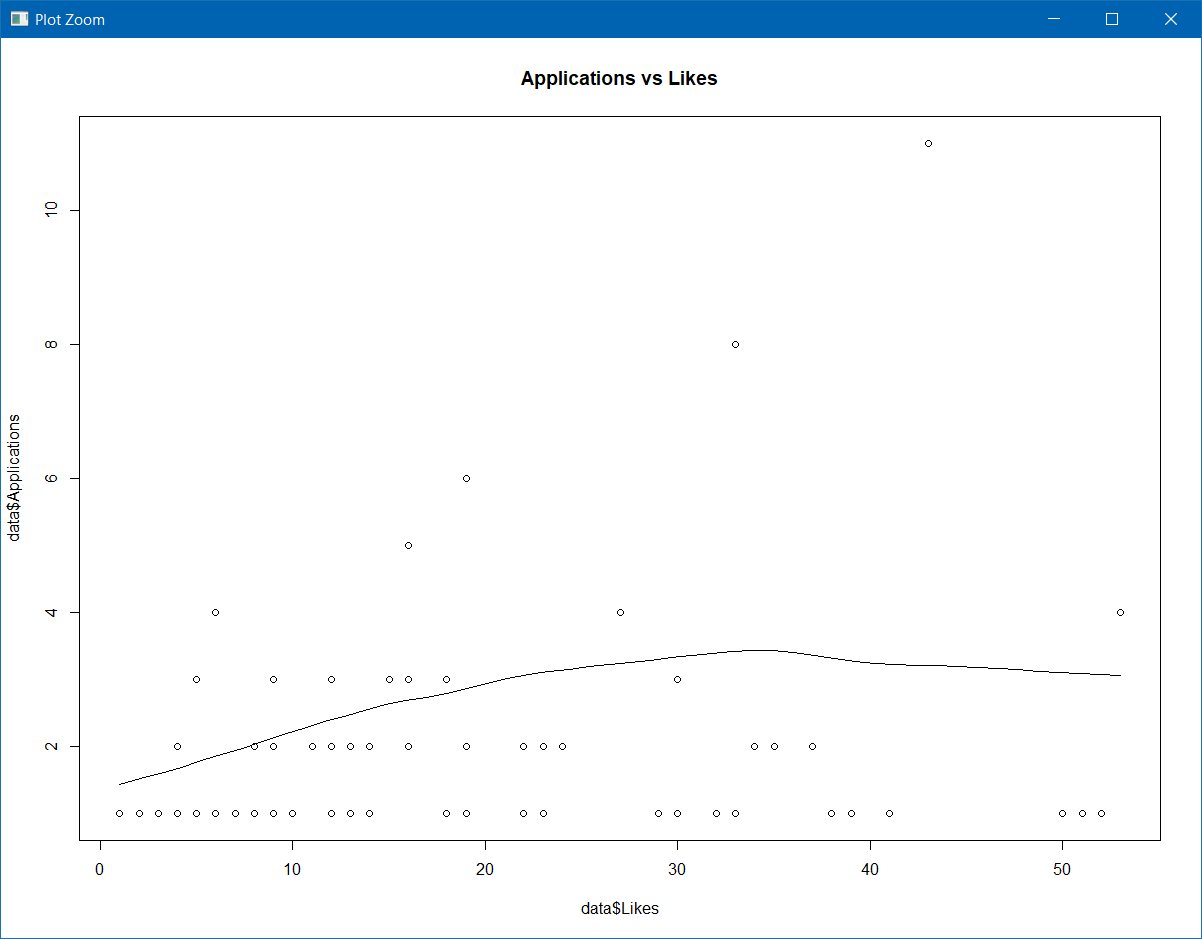


Рисунок 5 – График рассеяния по количеству лайков и заявок

В качестве тренировочных были взяты первые 70% данных, в то время как тестовыми стали оставшиеся 30%. Для проверки качества предсказания высчитываются средние отклонения предсказанного от тестовых данных.

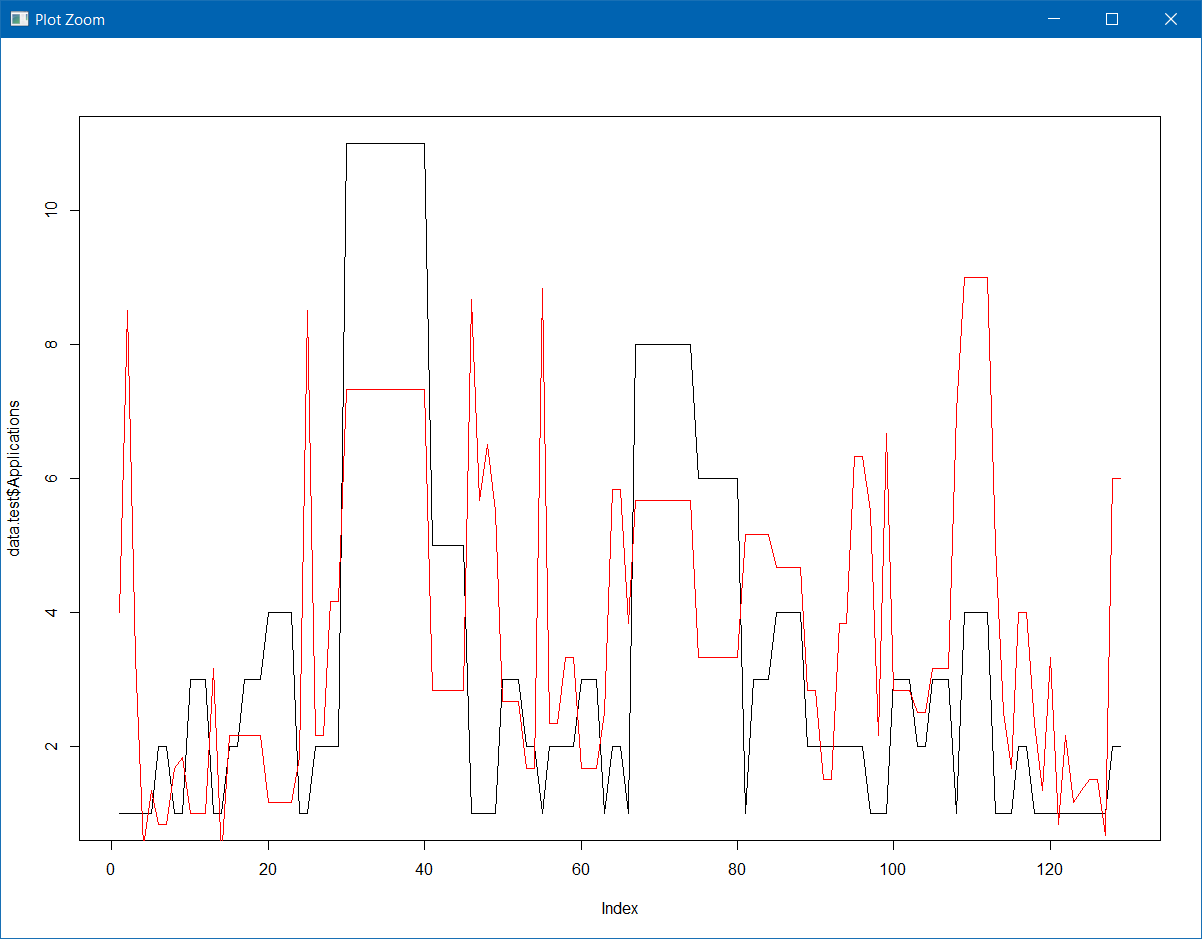


Рисунок 6 – График тестовых данных (чёрная линия) и предсказание модели (красная линия)

По результатам оценки модели показатель MAE = 2,197674, показатель p-value, рассчитанные функцией summary(model) также представлены на рисунке 7.

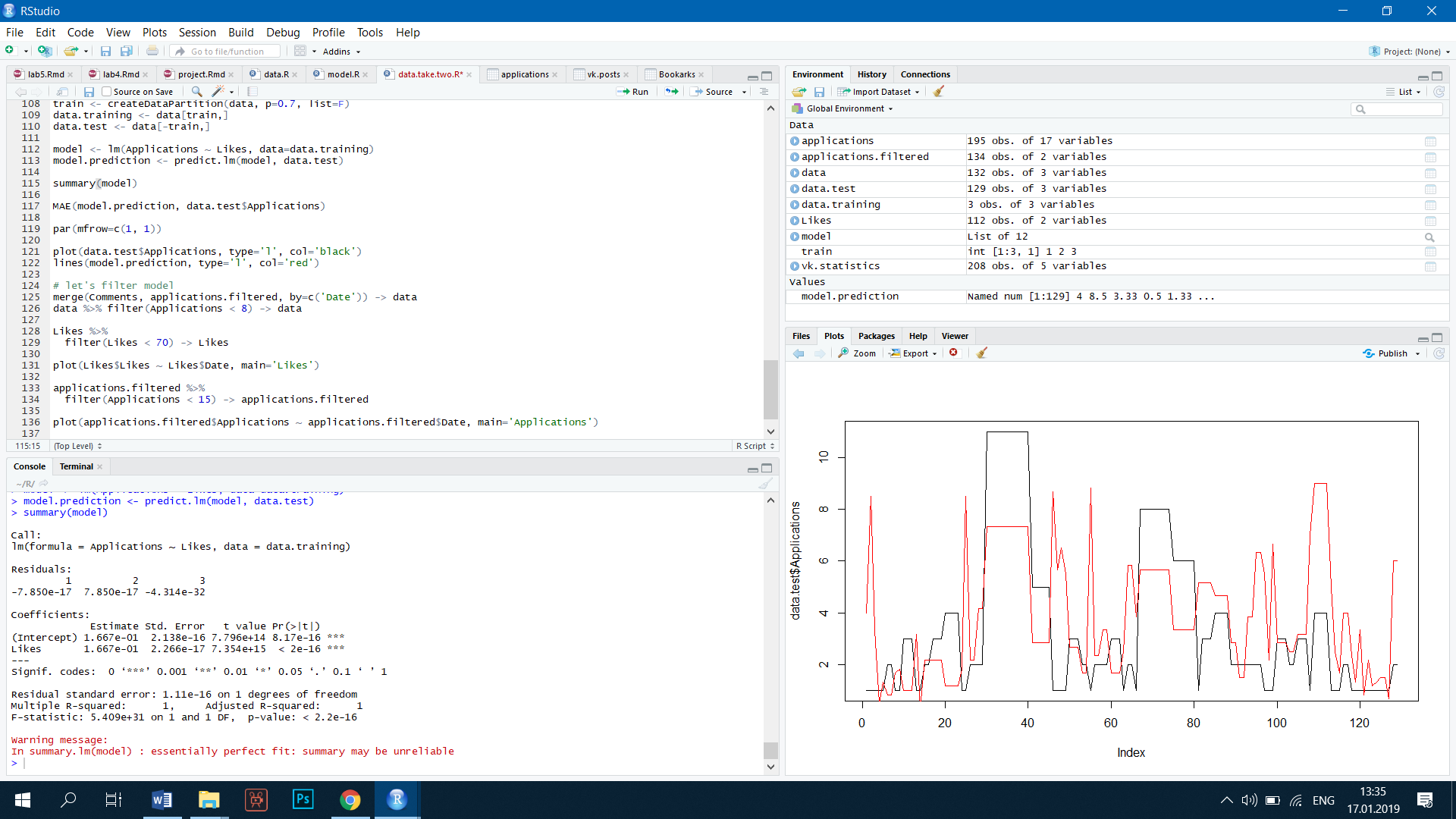


Рисунок 7 – Результат работы функции summary(model) на тренировочных данных

Показатели R-squared, Adjusted R-squared и p-value, рассчитанные на полном наборе данных приведены на рисунке 8.

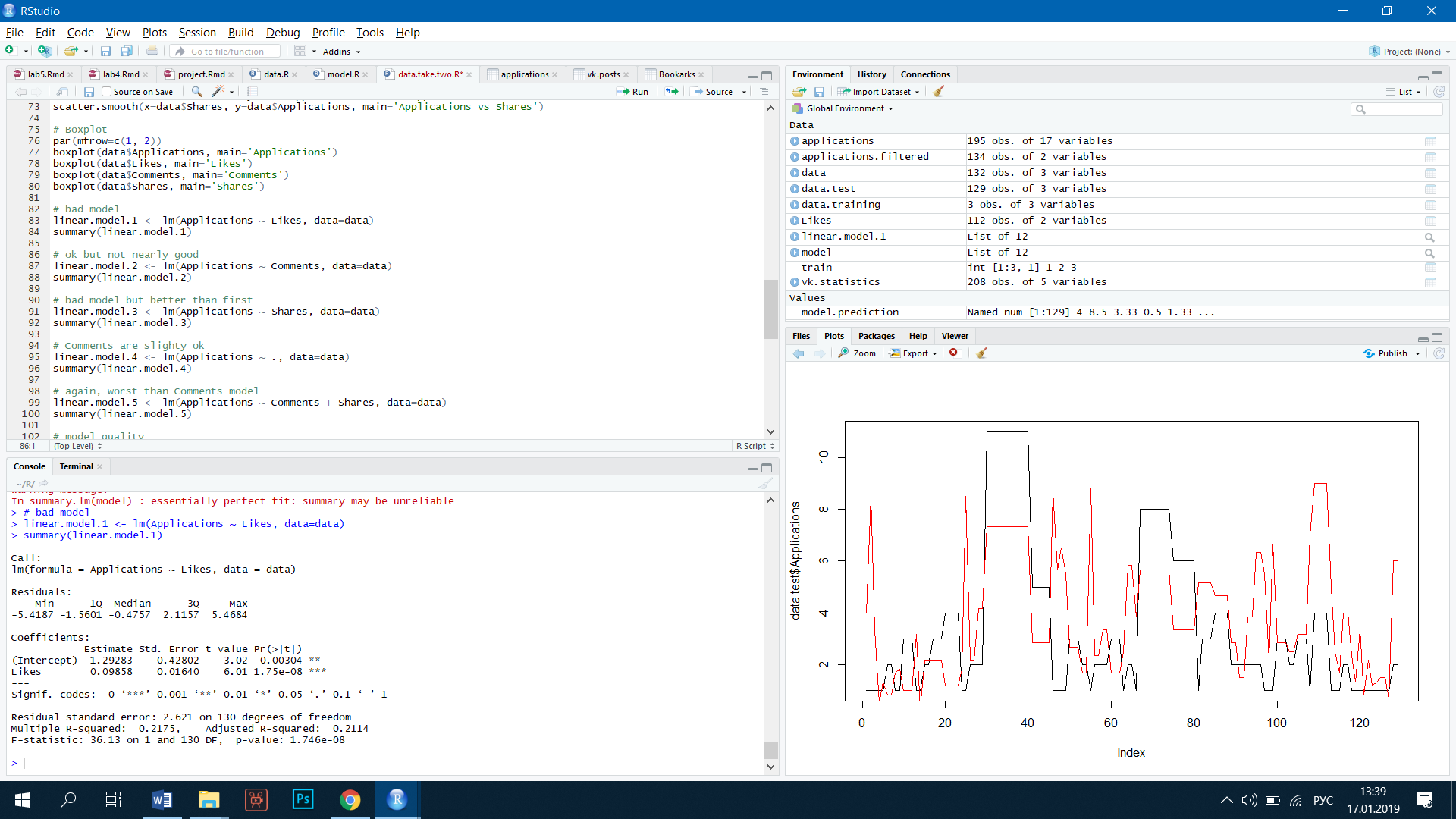


Рисунок 8 – Результат работы функции summary(model) на полном наборе данных

По результатам предсказания, можно сделать вывод, что существует незначительная зависимость между количество заявок и количеством лайков на странице компании в социальной сети.

# Разработка сервиса

Для реализации сервиса, через который пользователь будет пользоваться построенной аналитической моделью, был выбран язык программирования C# и Windows Forms.

Сервис позволяет конечному пользователю выбирать желаемое количество поступающих заявок (рис. 9), на котором и строится дальше модель. Данные берутся из R скрипта. Полный код R скрипта находится в приложении А.

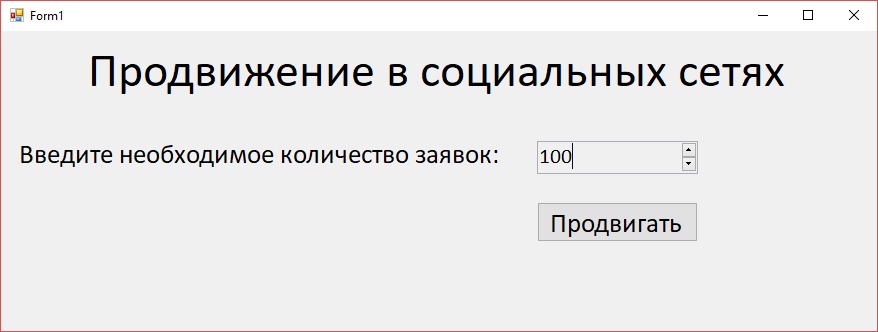


Рисунок 9 – Экран выбора количества заявок для модели

Когда нужно создать модель и получить её результат, сервис запускает R скрипт через подпроцесс и анализирует то, что ему процесс выдал в *stdout* и *stderr*.

На выходе мы можем увидеть предсказанное количество лайков, при котором мы получим желаемое количество заявок. Качество модели можно оценить в виде четырех показателей – p-value, R-squared, Adjusted R-squared и MAE.

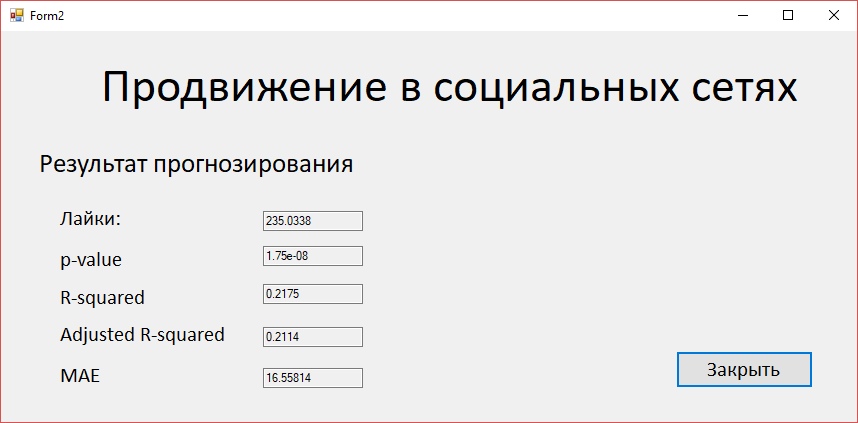


Рисунок 10 – Экран результата выполнения модели

Из-за выбора WinForms для разработки данного проекта появляется проблема с дистрибьюцией источника данных и самого R скрипта. Обновление данных возможно только с выпуском новой версии программы.

Основной код, где производится загрузка R скрипта на языке C# приведен в Приложении C.

# Заключение

Таким образом, в ходе выполнения проекта, были выгружены из социальной сети «Вконтакте» необходимые данные в *CSV* файлы, полученные наборы данных были обработаны и проанализированы. После построения необходимых моделей предположение о существовании зависимости между количеством заявок (*Applications*) и количеством лайков на странице организации (*Likes*) подтвердилась.

Для проверки качества модели был рассмотрены показатели*,* выдаваемые функцией summary(model), такие как p-value, R-squared и Adjusted R-squared, а также модульное отклонение MAE. По значениям данных показателей было установлено, что на данном наборе данных зависимость присутствует, однако она лишь приближена к идеалу. Возможно, увеличение количества данных при прогнозировании могло бы привести к более точному результату.

На финальном этапе работы было спроектировано консольное приложение на языке C# с использованием Windows Forms. Сервис позволяет конечному пользователю выбирать желаемое количество заявок, на котором и строится дальше модель, а также, на втором экране отображает предсказаное количество лайков и различные показатели для оценки полученной модели.

Обработка данных и построение моделей производится в файле data.take.three.R, код которого приведен в приложении А. Файл Model.R - финальная версия создания модели, выдача результата и метрик качества модели и предсказания, код данного файла приведен в приложении В.

Фрагмент кода, в котором производится загрузка R скрипта и вывод данных в WinForms на языке C# находится в файле Form2, код которого приведен в приложении С. Файл dataset.Rda – обработанный набор данных.

Полностью проект можно найти на GitHub по ссылке <https://github.com/diewdokimowa/MD-DA-2018/tree/master/R%20project>. Однако, исходные csv-файлы не опубликованы в связи с конфиденциальностью содержащихся в них данных.

# Библиографический список

1. Дубров А.М. Многомерные статистические методы /А.М. Дубров.– 2013. – 252 с.
2. Гришин А.Ф. Статистика: Учеб. пособие./ А.Ф. Гришин - 2003. – 240с.
3. Введение в многомерный анализ [Электронный ресурс]. Доступен по ссылке – https://habr.com/post/126810/ Последняя дата обращения: 25.11.18.
4. Кречетов Н. Продукты для интеллектуального анализа данных / Н. Кречетов, ComputerWeek-Москва. 2013 г. № 14-15. 32-39 с.
5. Вендров А.М. Практикум по проектированию программного обеспечения ЭИС: Учебное пособие / А.М. Вендров, Финансы и статистика, 2014. 215с.

# ПРИЛОЖЕНИЕ А

Исходный код на R (файл data.take.three.R)

library(caret)

library(spatstat)

library(lubridate)

library(dplyr)

applications <- read.csv2('applications.csv', header=TRUE, sep=';')

vk.statistics <- read.csv2('vk\_stat.csv' ,header=TRUE, sep=',')

summary(vk.statistics)

# Converting columns

names(applications) <- c(

"Date",

"Number",

"Status",

"Comment",

"id",

"User.first.name",

"User.last.name",

"Birthday",

"Sex",

"City",

"Country",

"First.name",

"Last.name",

"Phone",

"Mail",

"User",

"Link")

applications$Date <- as.Date(applications$Date, '%d.%m.%Y')

vk.statistics$Date <- as.Date(vk.statistics$Date, '%d.%m.%Y')

applications %>%

group\_by(Date=floor\_date(Date, "day")) %>%

mutate(Applications=n()) %>%

dplyr::select(Date, Applications) -> applications.filtered

vk.statistics %>%

filter(Parameter.1 == 'Likes') %>%

mutate(Likes=Value) %>%

dplyr::select(Date, Likes) -> Likes

vk.statistics %>%

filter(Parameter.1 == 'Comments') %>%

mutate(Comments=Value) %>%

dplyr::select(Date, Comments) -> Comments

vk.statistics %>%

filter(Parameter.1 == 'Shares') %>%

mutate(Shares=Value) %>%

dplyr::select(Date, Shares) -> Shares

merge(applications.filtered, Likes, by=c('Date')) -> data

merge(data, Comments, by=c('Date')) -> data

merge(data, Shares, by=c('Date')) -> data

par(mfrow=c(1,1))

plot(data$Applications ~ data$Date, main='Applications')

plot(data$Comments ~ data$Date, main='Comments')

plot(data$Likes ~ data$Date, main='Likes')

plot(data$Shares ~ data$Date, main='Shares')

# Scatter

par(mfrow=c(1, 1))

scatter.smooth(x=data$Likes, y=data$Applications, main='Applications vs Likes')

scatter.smooth(x=data$Comments, y=data$Applications, main='Applications vs Comments')

scatter.smooth(x=data$Shares, y=data$Applications, main='Applications vs Shares')

# Boxplot

par(mfrow=c(1, 2))

boxplot(data$Applications, main='Applications')

boxplot(data$Likes, main='Likes')

boxplot(data$Comments, main='Comments')

boxplot(data$Shares, main='Shares')

# bad model

linear.model.1 <- lm(Applications ~ Likes, data=data)

summary(linear.model.1)

# ok but not nearly good

linear.model.2 <- lm(Applications ~ Comments, data=data)

summary(linear.model.2)

# bad model but better than first

linear.model.3 <- lm(Applications ~ Shares, data=data)

summary(linear.model.3)

# Comments

linear.model.4 <- lm(Applications ~ ., data=data)

summary(linear.model.4)

# again, worst than Comments model

linear.model.5 <- lm(Applications ~ Comments + Shares, data=data)

summary(linear.model.5)

# model quality

summary(linear.model.2)

AIC(linear.model.2)

BIC(linear.model.2)

# let's test model

train <- createDataPartition(data, p=0.7, list=F)

data.training <- data[train,]

data.test <- data[-train,]

model <- lm(Applications ~ Likes, data=data.training)

model.prediction <- predict.lm(model, data.test)

summary(model)

MAE(model.prediction, data.test$Applications)

par(mfrow=c(1, 1))

plot(data.test$Applications, type='l', col='black')

lines(model.prediction, type='l', col='red')

# let's filter model

merge(Comments, applications.filtered, by=c('Date')) -> data

data %>% filter(Applications < 8) -> data

Likes %>%

filter(Likes < 70) -> Likes

plot(Likes$Likes ~ Likes$Date, main='Likes')

applications.filtered %>%

filter(Applications < 15) -> applications.filtered

plot(applications.filtered$Applications ~ applications.filtered$Date, main='Applications')

save(data, file='dataset.Rda')

# ПРИЛОЖЕНИЕ B

Исходный код R скрипта (файл model.R)

library(caret)

library(optparse)

library(dplyr)

option\_list <- list(

make\_option(c('--applications'), action='store', default=NA, type='double')

)

opt = parse\_args(OptionParser(option\_list=option\_list))

load('dataset.Rda')

linear.model <- lm(Likes ~ Applications, data=data)

# grab p-value, R-squared and Adjusted R-squared

summary(linear.model)

# print prediction result (how much likes should be under the posts per day)

predict(linear.model, data.frame(Applications=opt$applications))[[1]]

# model testing

train <- createDataPartition(data, p=0.7, list=F)

data.training <- data[train,]

data.test <- data[-train,]

model <- lm(Likes ~ Applications, data=data.training)

model.prediction <- predict(model, data.test)

# show in interface, smaller number is better

# Mean Absolute Error

MAE(model.prediction, data.test$Applications)

# ПРИЛОЖЕНИЕ С

Код приложения на C# (файл Form2.cs)

|  |
| --- |
| using System; |
|  | using System.Collections.Generic; |
|  | using System.ComponentModel; |
|  | using System.Data; |
|  | using System.Diagnostics; |
|  | using System.Drawing; |
|  | using System.Linq; |
|  | using System.Text; |
|  | using System.Threading.Tasks; |
|  | using System.Windows.Forms; |
|  |  |
|  | namespace WindowsFormsApp1 |
|  | { |
|  | public partial class Form2 : Form |
|  | { |
|  | private decimal applications; |
|  |  |
|  | public Form2(decimal applications) |
|  | { |
|  | this.applications = applications; |
|  | InitializeComponent(); |
|  |  |
|  | runModel(); |
|  | } |
|  |  |
|  | private void runModel() |
|  | { |
|  | var process = new Process(); |
|  | process.StartInfo.FileName = "Rscript"; |
|  | process.StartInfo.Arguments = $"--vanilla ../../model.R --applications {applications}"; |
|  | process.StartInfo.UseShellExecute = false; |
|  | process.StartInfo.RedirectStandardOutput = true; |
|  | process.StartInfo.RedirectStandardError = true; |
|  | process.StartInfo.WindowStyle = ProcessWindowStyle.Minimized; |
|  |  |
|  | process.Start(); |
|  | process.WaitForExit(); |
|  | var result = process.StandardOutput.ReadToEnd(); |
|  | var error = process.StandardError.ReadToEnd(); |
|  | if (string.IsNullOrEmpty(result)) |
|  | { |
|  | MessageBox.Show(error, "Произошла ошибка", MessageBoxButtons.OK); |
|  | return; |
|  | } |
|  | parseOutput(result); |
|  | } |
|  |  |
|  | private void parseOutput(string output) |
|  | { |
|  | var lines = output.Split('\n'); |
|  |  |
|  | var line11 = lines[11].Trim().Split(' '); |
|  | pValue.Text = line11[line11.Length - 2]; |
|  |  |
|  | var line16 = lines[16].Trim().Split(','); |
|  | rSquaredValue.Text = line16[0].Split(' ').Last(); |
|  | adjustedRSquaredValue.Text = line16[1].Split(' ').Last(); |
|  |  |
|  | var line19 = lines[19].Trim().Split(' '); |
|  | commentsValue.Text = line19.Last(); |
|  |  |
|  | var line20 = lines[20].Trim().Split(' '); |
|  | maeValue.Text = line20.Last(); |
|  | } |
|  |  |
|  | private void button1\_Click(object sender, EventArgs e) |
|  | { |
|  | Close(); |
|  | } |
|  | } |
|  | } |