Министерство науки и высшего образования 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 работы

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члены комиссии

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

РАЗРАБОТКА СИСТЕМЫ ПРОГНОЗИРОВАНИЯ СТОИМОСТИ ДОМА

Пояснительная записка к курсовому проекту по модулю

«Методы анализа Big Data»

Преподаватель Мирвода С.Г.

Студенты Куштанов Д.И.

Изотов И.Н.

Группа РИ-450004

Екатеринбург

2019

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc534994740)

[1 ОПИСАНИЕ НАБОРА ДАННЫХ 4](#_Toc534994741)

[2 ТРЕНИРОВКА ЛИНЕЙНОЙ МОДЕЛИ 7](#_Toc534994742)

[3 ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ 11](#_Toc534994743)

[ЗАКЛЮЧНИЕ 13](#_Toc534994744)

[БИБЛИОГРАФИЧЕСКИЙ СПИСОК 14](#_Toc534994745)

[ПРИЛОЖЕНИЕ А – КОД RMARDOWN 15](#_Toc534994746)

[ПРИЛОЖЕНИЕ Б – КОД ФУННКЦИИ R 17](#_Toc534994747)

[ПРИЛОЖЕНИЕ В – КОД ОСНОВНОЙ ФУНКЦИИ PYTHON 18](#_Toc534994748)

# ВВЕДЕНИЕ

В текущее время объемы информации растут по экспоненте. Для того, чтобы быстрее реагировать на изменения рынка, получить конкурентные преимущества и повысить эффективность производства, нужно получить, обработать и проанализировать огромное количество данных. Для этого используются специальные программные средства, и одним из таких решений может являться язык программирования для статистической обработки данных и работы с графикой R.

Целью данной работы является разработка системы прогнозирования стоимости домов, с использованием открытого набора данных.

Гипотезой служит предположение о том, что технические и геолокационные характеристики дома вносят влияние на формирование его стоимости.

Для достижения поставленной цели необходимо выполнить следующие задачи:

1. изучить набор данных;
2. провести анализ данных;
3. натренировать линейную модель;
4. написать пользовательское приложение.

# 1 ОПИСАНИЕ НАБОРА ДАННЫХ

Для данный работы использовался набор данных стоимостей домов США в определенном регионе [1]. Пример данных изображен на рисунке 1.

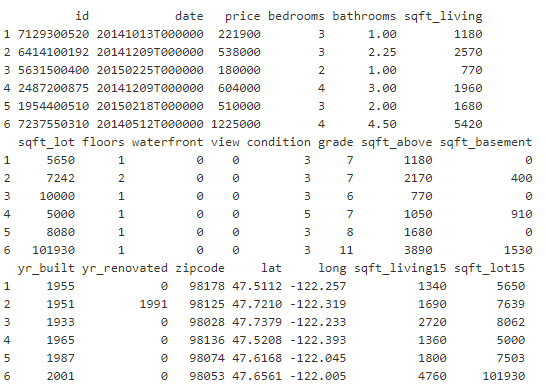


Рисунок 1 – Пример данных

В данном наборе данных колонки имеют следующие значения:

* id – номер дома;
* date – дата продажи дома;
* price – цена;
* bedrooms – число спален;
* bathrooms – число ванных комнат;
* sqft\_living – площадь дома;
* sqft\_lot – площадь участка;
* floors – число этажей;
* waterfront – есть ли выход на набережную;
* view – был ли осмотрен;
* condition – состояние дома;
* grade – оценка дома основанная на King County;
* sqft\_above – площадь дома отдельно от подвала;
* sqft\_basement – площадь подвала;
* yr\_built – год постройи;
* yr\_renovated – год ремонта дома;
* zipcode – почтовый zip код;
* lat – координата широты;
* long – координата долготы;
* sqft\_living15 – жилая площадь на 2015 год;
* sqft\_lot15lot – площадь участка на 2015 год.

На рисунках ниже (рисунки 2, 3, 4, 5) отображены некоторые зависимости атрибутов от цены.

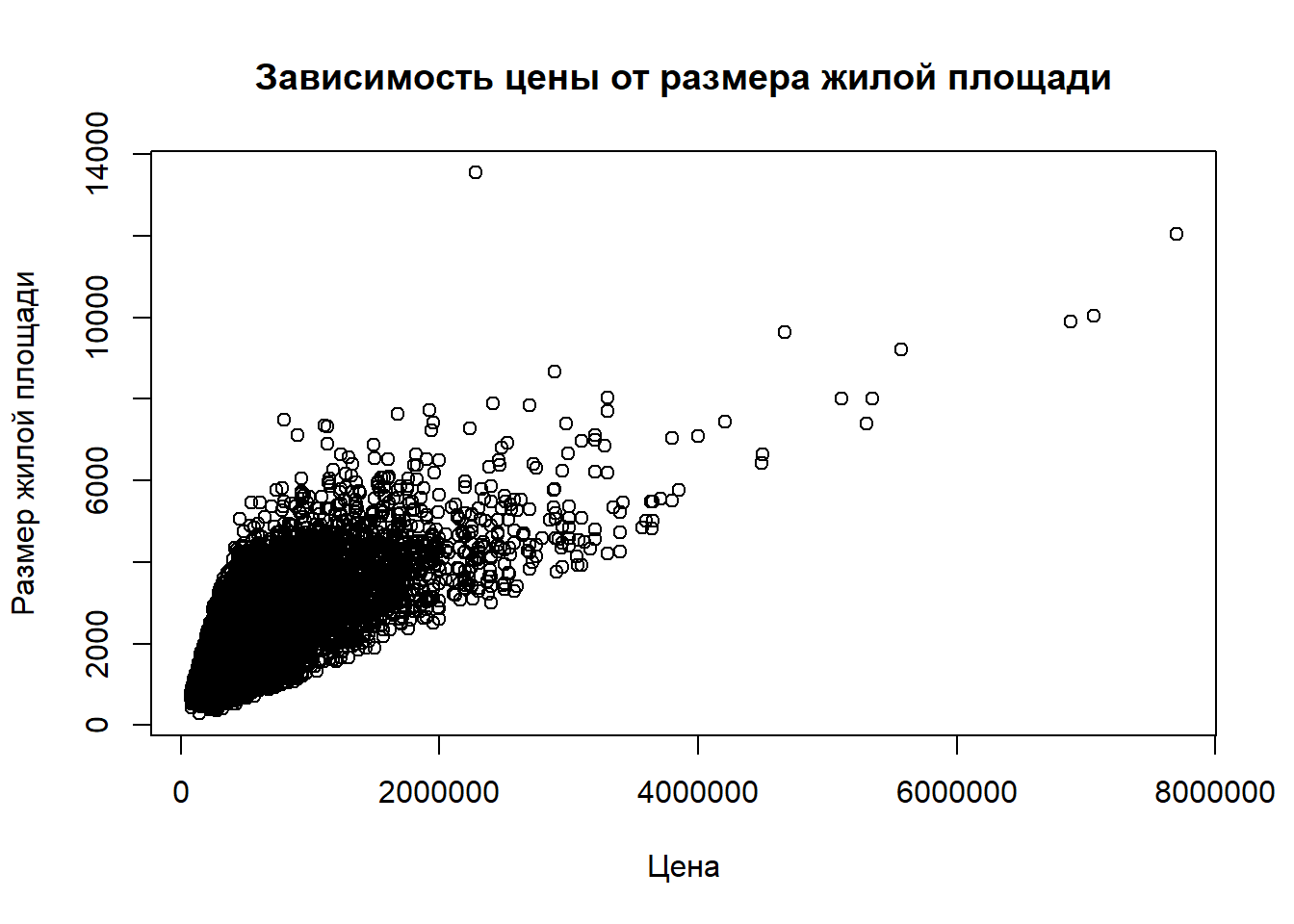


Рисунок 2 – Зависимость цены от размера жилой площади

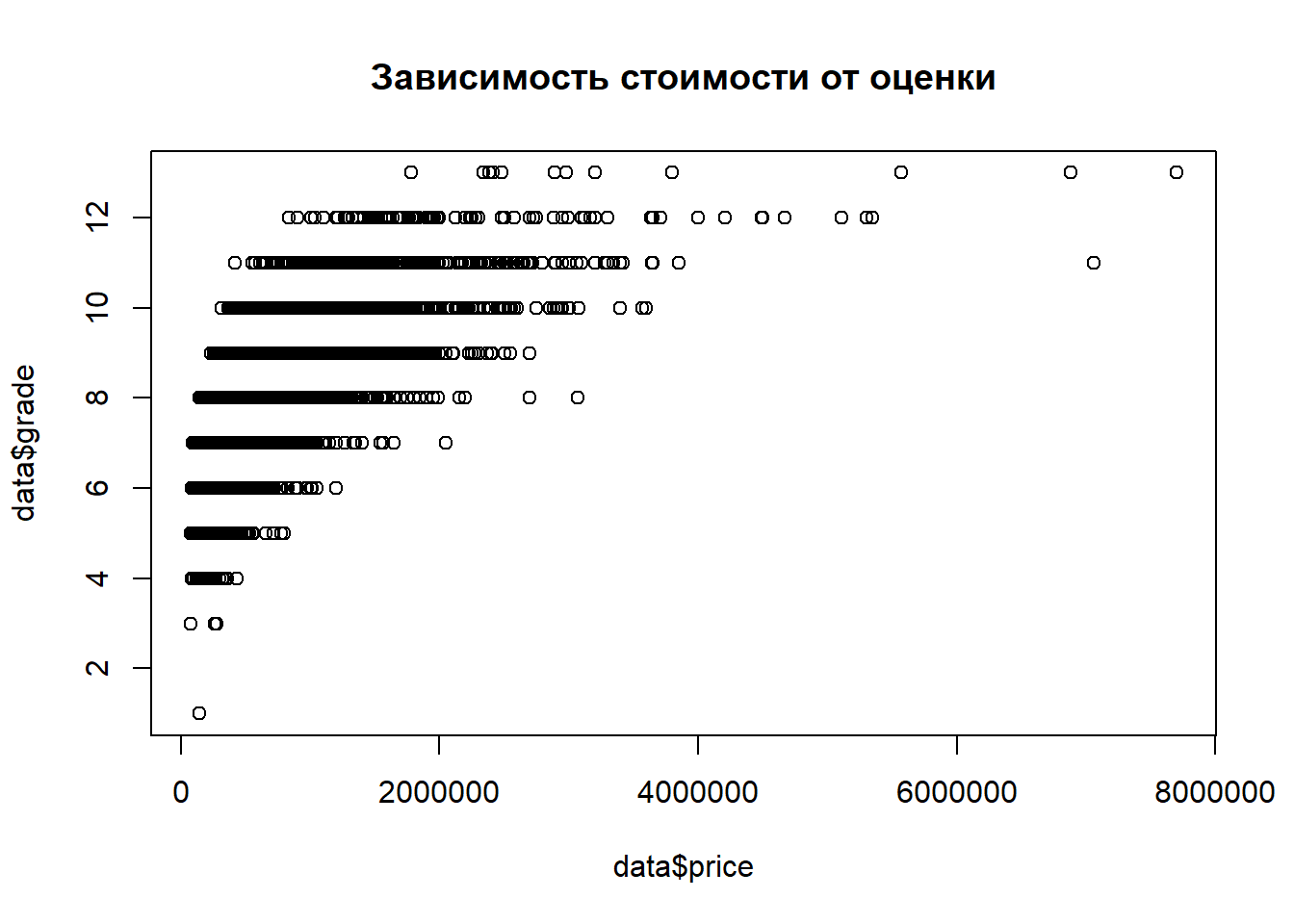


Рисунок 3 – Зависимость цены от оценки

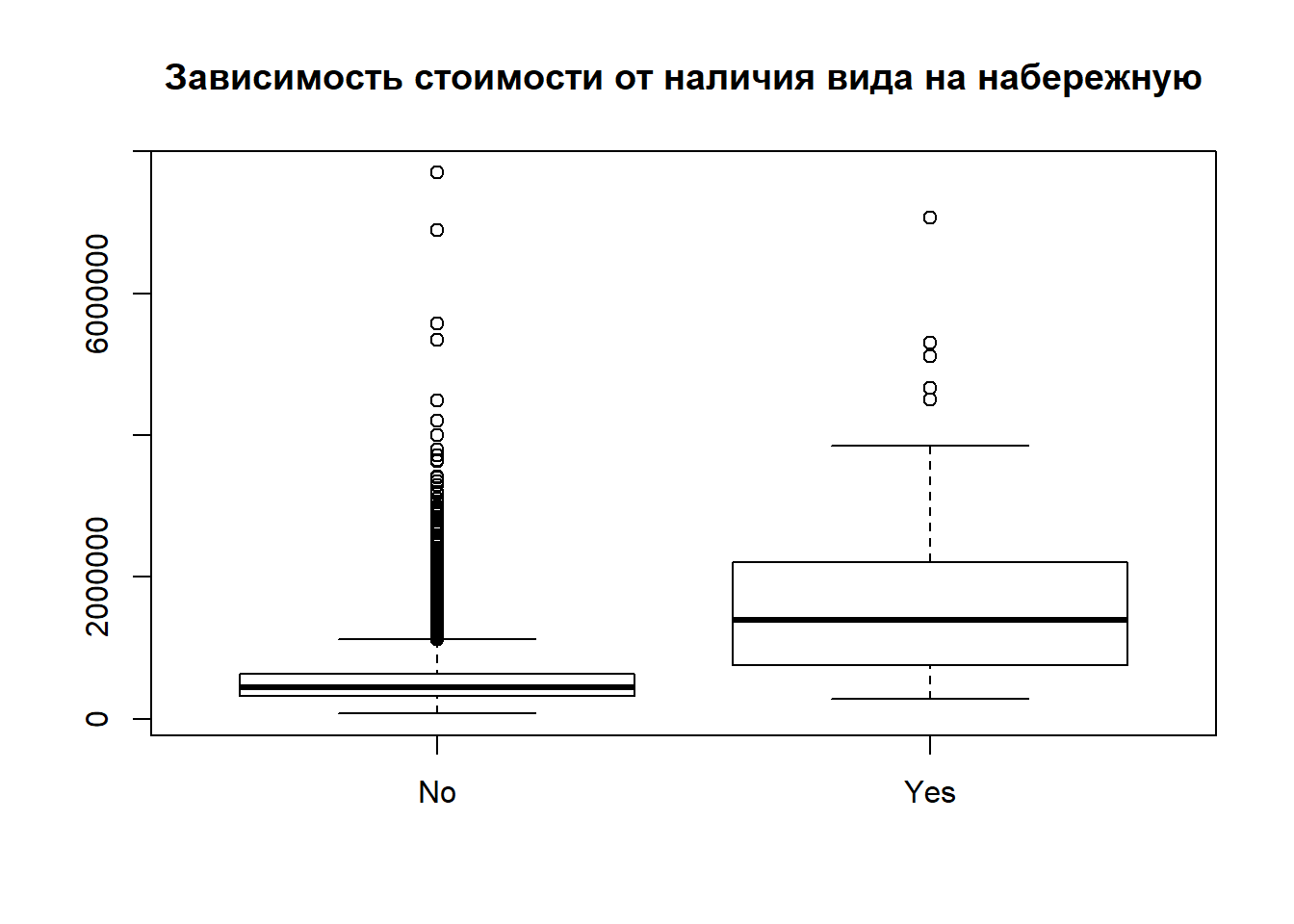


Рисунок 4 – Зависимость цены от наличия вида на набережную

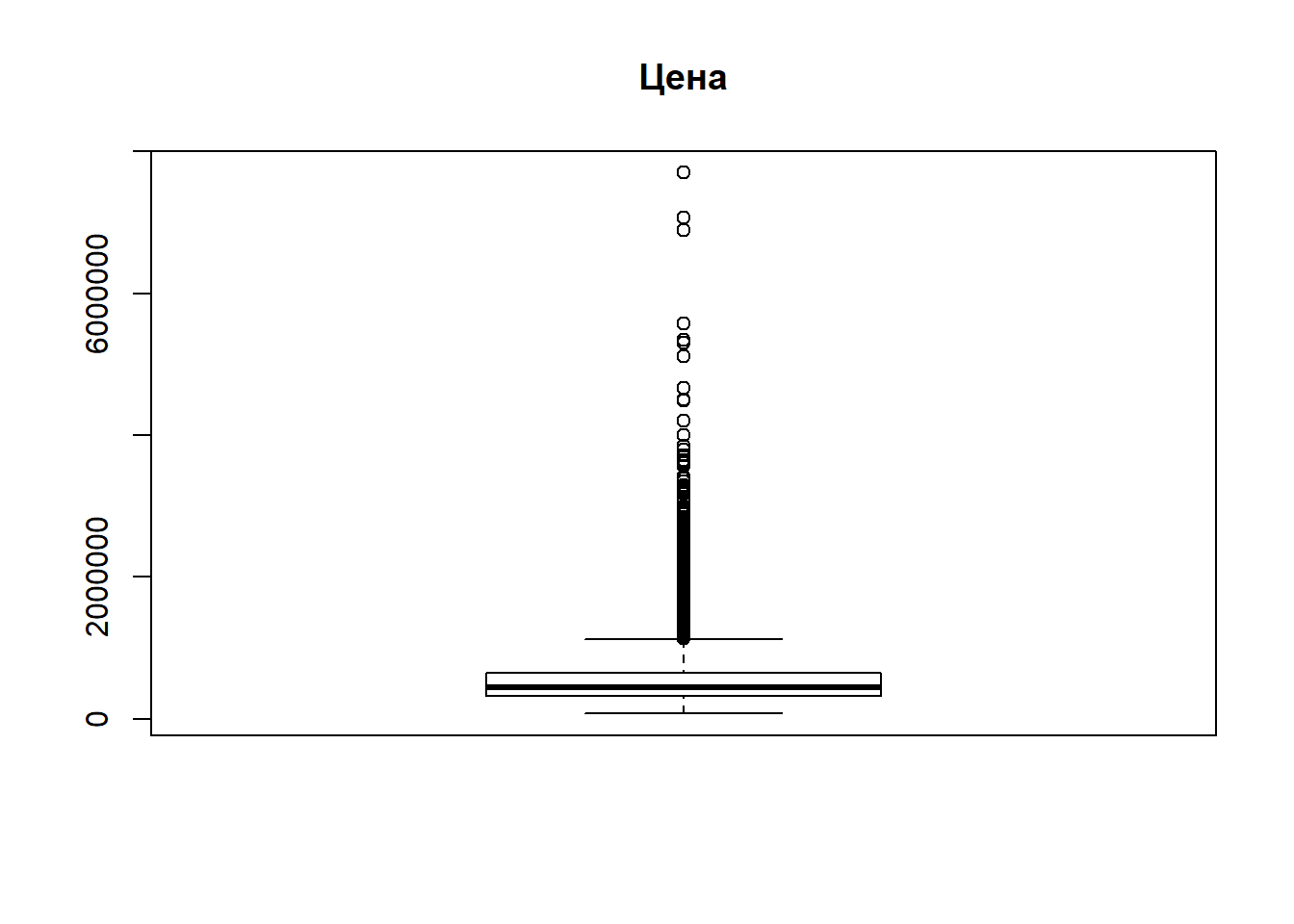


Рисунок 5 – График стоимости

# 2 ТРЕНИРОВКА ЛИНЕЙНОЙ МОДЕЛИ

Для тренировки модели была проведена чистка данных (результат на рисунке 6) и построена линейная модель на основе всех данных, чтобы определить наиболее значимые параметры (рисунок 7).



Рисунок 6 – Очищенная зависимость стоимости от вида на набережную

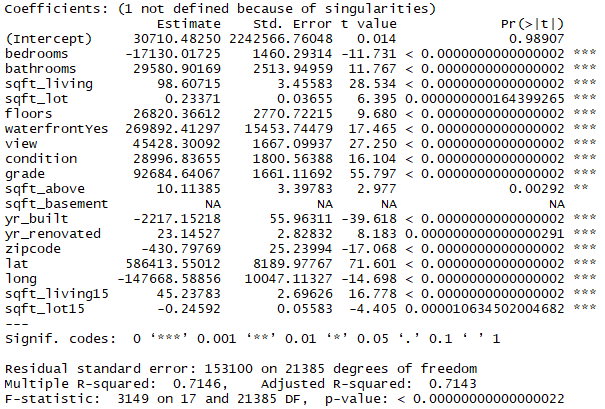


Рисунок 7 – Зависимость влияния параметров на цену

Для построения более точной модели были убраны наименее значимые столбцы и создан дополнительный avgZipCodePrice являющийся средней ценой по почтовому индексу. Результаты показаны на рисунках 8, 9, 10.

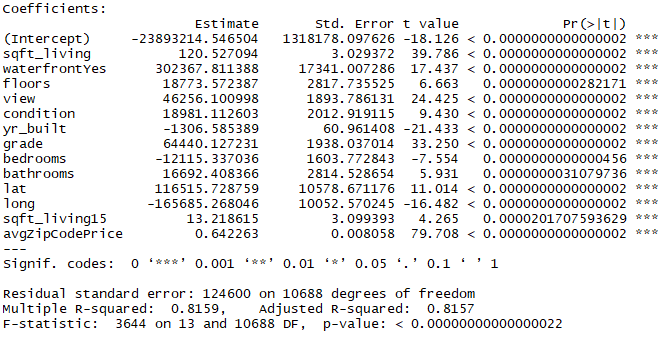


Рисунок 8 – Коэффициенты новой модели

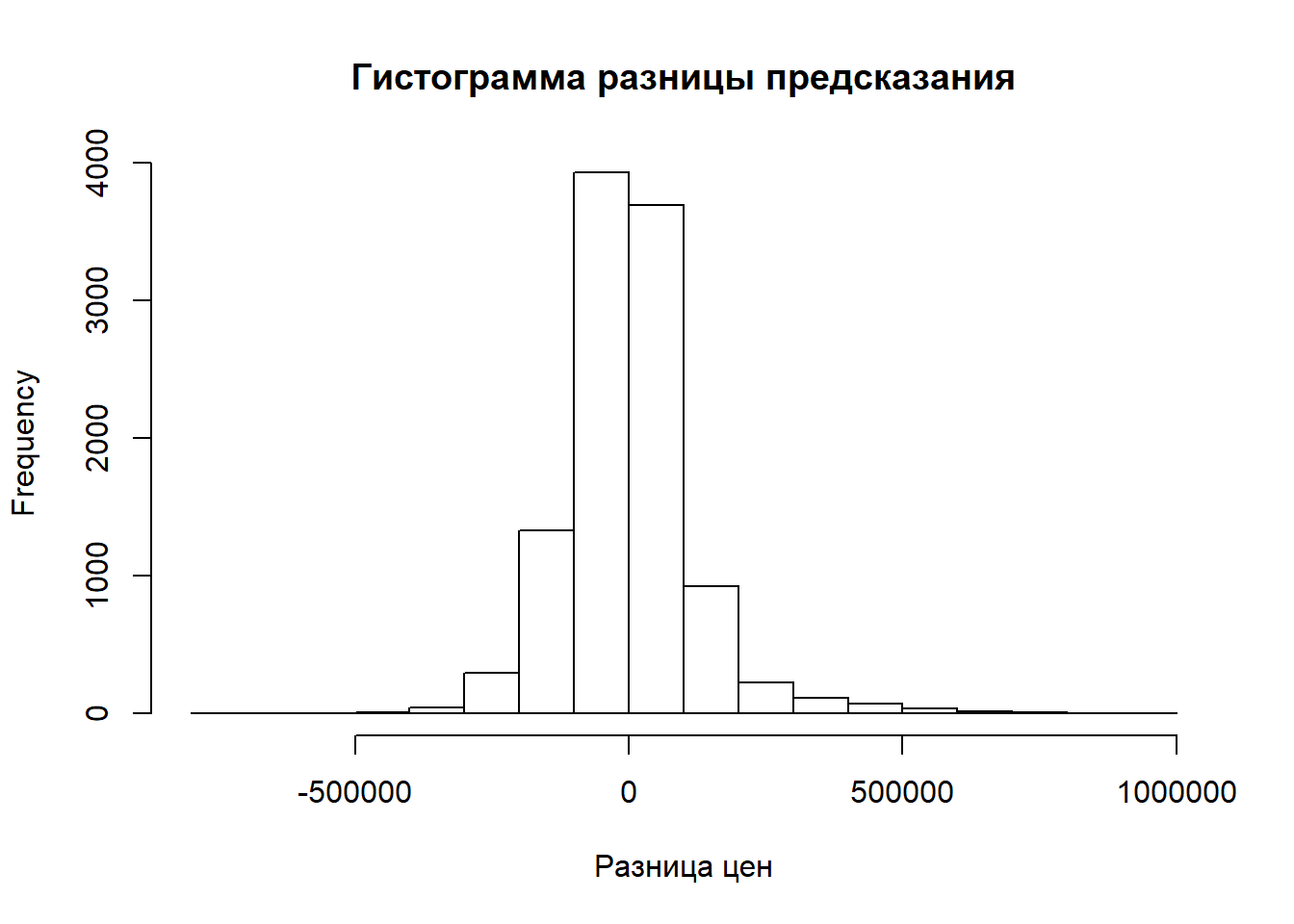


Рисунок 9 – Гистограмма зависимости предсказанных значений от проверяемых

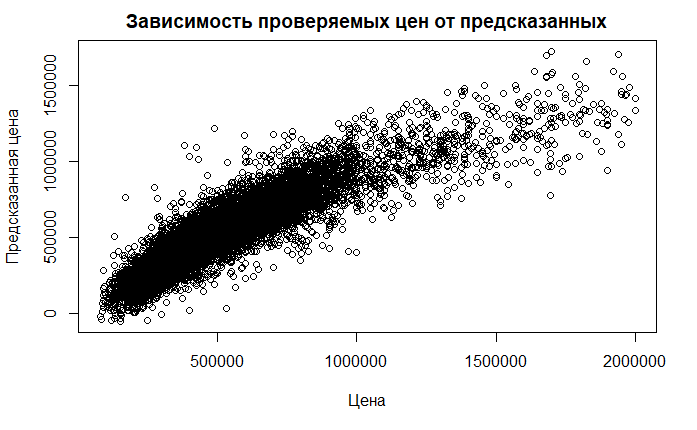


Рисунок 10 – Зависимость предсказанных значений от проверяемых

Для приложения было принято решение использовать меньшее число параметров, для этого была создана дополнительная модель. Ее результат показан на рисунках 11, 12, 13.

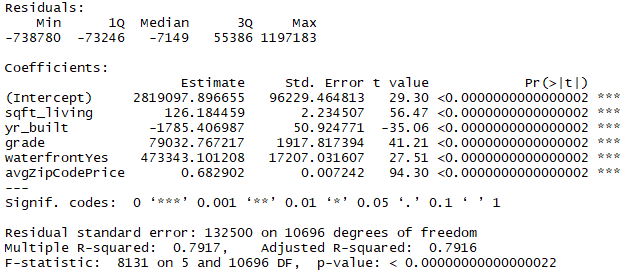


Рисунок 11 – Коэффициенты модели с меньшим числом параметров



Рисунок 12 – Гистограмма зависимости предсказанных значений от проверяемых модели с меньшим числом параметров



Рисунок 13 – Зависимость предсказанных значений от проверяемых модели с меньшим числом параметров

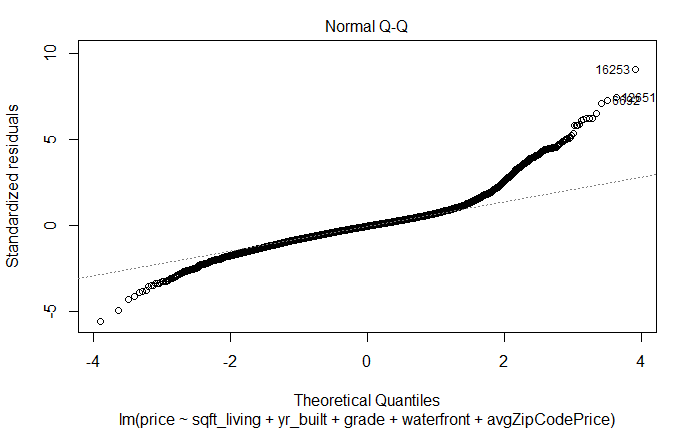


Рисунок 14 – Проверка зависимости при помощи графика Q-Q

# 3 ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ

Приложение было написано на языке Python. Его интерфейс изображен на рисунке 12.

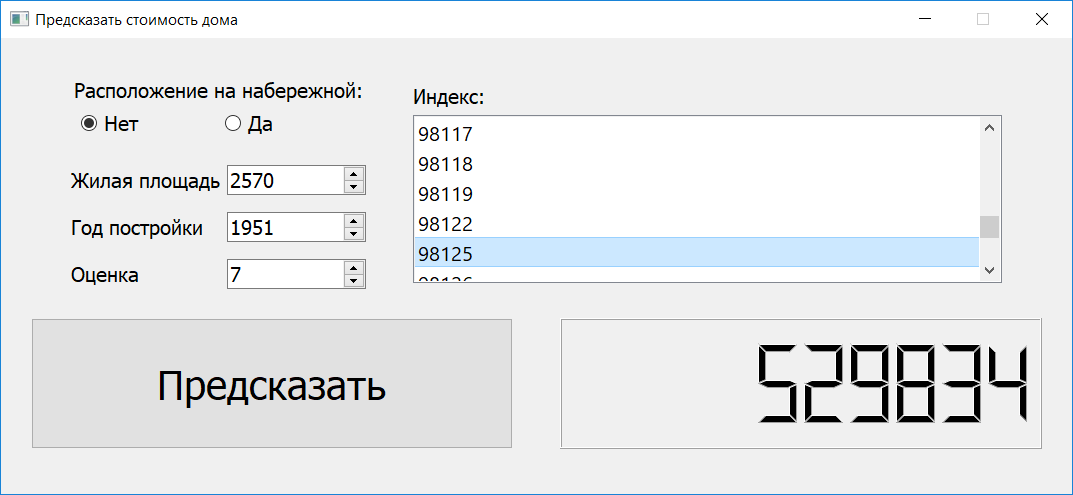


Рисунок 15 – Макет приложения

Результат вызова функции из R с теми же параметрами, что представлены на изображение макета, показан на рисунке 13. Результаты вывода у них совпадают.



Рисунок 16 – Вызов функции предсказания в R

Интерфейс приложения создавался с использованием библиотеки PyQt5 с помощью инструмента Qt Designer. Для получения возможности использования языка R была использована библиотека rpy2.

Приложение при инициализации получает значения почтовых индексов из набора данных, обработанных языком R. При нажатии на кнопку «Предсказать» данные из формы передаются в функцию R для расчета цены дома. Полный код программы, ее интерфейс и функция на языке R расположены в приложении Б и В.

**Краткая инструкция пользователя:**

1. Запустить приложение;
2. Ввести требуемые параметры в окна ввода (возможные вводимые параметры ограничены граничными значениями). Выбрать необходимый почтовый индекс;
3. Нажить на кнопку «Предсказать»;
4. Получить значение в окне справа от нажимаемой кнопки.

# ЗАКЛЮЧНИЕ

В результате выполнения данной работы был изучен открытый набор данных стоимости домов, путем изучения представленной информации на сайте источника и отображением графиков и параметров при помощи функций языка программирования R в программе Rstudio. Была построена линейная модель по этим данным для предсказания стоимости домов. При этом использовался язык Rmarkdown в котором были построены возможные зависимости между ценой и другими параметрами, а также проведена очистка от выбросов и изучены значения, наиболее влияющие факторы на цену дома. Значение для оптимального числа параметров в линейной модели составило 0,82.

Для отображения возможного применения данных расчетов было реализовано приложение языке Python, в котором реализована функциональность в виде обращения к функциям языка R для применения и расчета линейной модели. В данном случае использовалась линейная модель с меньшим числом параметров, для которой .

Также на языке python было разработано приложение, упрощающее взаимодействие пользователя с линейной моделью.

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. House Sales in King County, USA. Predict house price using regression [Электронный ресурс]: [веб-сайт]. – Kaggle. – Режим доступа: https://www.kaggle.com/harlfoxem/housesalesprediction, свободный. (Дата обращения: 11.01.2019)
2. Qt Documentation Manual [Электронный ресурс]: [веб-сайт]. – Qt. – Режим доступа: http://doc.qt.io/Qt-5/qtdesigner-manual.html, свободный (Дата обращения: 11.01.2019)

# ПРИЛОЖЕНИЕ А – КОД RMARDOWN

---

title: "Курсовая работа"

author: "Изотов Илья, Куштанов Дмитрий РИ-450004"

---

```{r}

options(scipen=10000)

data <- read.csv("kc\_house\_data.csv", header=TRUE, sep=",")

head(data)

summary(data)

data <- data[,-c(1,2)]

data$waterfront <- factor(data$waterfront, levels = c(0,1))

levels(data$waterfront) <- c("No", "Yes")

hist(data$price, xlab = "Цена", main = "Цена")

hist(data$sqft\_living, xlab = "Размер жилой площади", main = "Жилая площадь")

hist(data$yr\_built, xlab = "Год постройки", main = "Год постройки")

```

###Визулизируем возможные зависимости

```{R}

plot(data$price, data$sqft\_living,'p',main = "Зависимость стоимости от размера жилой площади")

plot(data$price, data$grade,'p',main = "Зависимость стоимости от оценки")

plot(data$waterfront, data$price,'p',main = "Зависимость стоимости от наличия вида на набережную")

boxplot(data$price, main = "Цена")

```

###Очистка от выбросов

```{r}

summary(data[data$waterfront=="No", ]$price)

data <- data[data$sqft\_living<13000,]

data <- data[(data$price<2\*10^6),]

data <- data[(data$grade<=12 & data$grade>3),]

plot(data$waterfront, data$price,'p',main = "Зависимость стоимости от наличия вида на набережную")

plot(sort(data$price), main = "Цена")

```

###Нахождение значимых параметров

```{r}

boxplot(data$price, main = "Цена")

lm.data <- lm(price~., data = data)

summary(lm.data)

plot(lm.data)

```

```{R}

plot(x= data$price, xlab = "Цена", ylab = "Размер жилой площади", y = data$sqft\_living,'p',main = "Зависимость стоимости от размера жилой площади")

plot(x= data$price,xlab = "Цена", ylab = "Оценка", y =data$grade,'p',main = "Зависимость стоимости от оценки")

```

###Добавление дополнительного параметра средней цены по почтовому индексу и разделение данных

```{r}

data$avgZipCodePrice <- ave(data$price,data$zipcode)

odds <- seq(1, nrow(data), by=2)

data.in <- data[odds, ]

data.out <- data[-odds, ]

```

###Тренировка первой модели

```{r}

lm.data.in <- lm(price~sqft\_living+waterfront+floors+view+condition+yr\_built+grade+bedrooms+bathrooms+lat+long+sqft\_living15+avgZipCodePrice, data = data.in)

data.predict.out <- predict(lm.data.in, data.out)

cor(data.out$price, data.predict.out)

summary(lm.data.in)

plot(x = data.out$price, xlab = "Цена", ylab = "Предсказанная цена",y = data.predict.out, main = "Зависимость проверяемых цен от предсказанных")

hist(data.out$price - data.predict.out, xlab = "Разница цен", main = "Гистограмма разницы предсказания")

```

###Уменьшение числа параметров для тренировки второй модели

```{r}

lm.data.in1 <- lm(price~sqft\_living+yr\_built+grade+waterfront+avgZipCodePrice, data = data.in)

data.predict.out1 <- predict(lm.data.in1, data.out)

cor(data.out$price, data.predict.out1)

summary(lm.data.in1)

plot(x = data.out$price, xlab = "Цена", ylab = "Предсказанная цена",y = data.predict.out1, main = "Зависимость проверяемых цен от предсказанных")

hist(data.out$price - data.predict.out1, xlab = "Разница цен", main = "Гистограмма разницы предсказания")

plot(lm.data.in1)

```

# ПРИЛОЖЕНИЕ Б – КОД ФУННКЦИИ R

# Функция получения списка потовых индексов

# Outputs: Возвращает уникальные номера почтовых индексов

# Для выполнения необходимо нахождение используемового набора данных в текущей директории

fun.zip<-function()

{

data <- read.csv("kc\_house\_data.csv", header=TRUE, sep=",");

data <- data[,-c(1,2)];

data <- data[data$sqft\_living<13000,];

data<- data[data$price<2\*10^6,];

data <- data[(data$grade<=12 & data$grade>3),]

data$avgZipCodePrice <- ave(data$price,data$zipcode);

return (unique(data$zipcode));

}

# Функция предсказания цены

# Inputs: sqft\_living = площадь жилья

# yr\_built = год постройки здания

# grade = оценка здания

# waterfront = Наличие выхода к воде (0 - нет, 1 - да)

# zipcode = почтовый индекс

# Outputs: Возвращает предсказанную цену

# Для выполнения необходимо нахождение набора данных в текущей директории

fun.predict<-function(sqft\_living, yr\_built, grade, waterfront, zipcode){

data <- read.csv("kc\_house\_data.csv", header=TRUE, sep=",");

data <- data[,-c(1,2)];

data <- data[data$sqft\_living<13000,];

data<- data[data$price<2\*10^6,];

data <- data[(data$grade<=12 & data$grade>3),]

data$avgZipCodePrice <- ave(data$price,data$zipcode)

avg <-mean(data[data$zipcode==zipcode,]$price)

lm.data.in <- lm(price~sqft\_living+yr\_built+grade+waterfront+avgZipCodePrice, data = data)

ret <- coef(lm.data.in)[1] + coef(lm.data.in)[2]\*sqft\_living +

coef(lm.data.in)[3]\*yr\_built + coef(lm.data.in)[4]\*grade +

coef(lm.data.in)[5]\*waterfront + coef(lm.data.in)[6]\*avg;

return (as.integer(ret))

}

# ПРИЛОЖЕНИЕ В – КОД ОСНОВНОЙ ФУНКЦИИ PYTHON

Файл main.py:

import sys

from PyQt5 import QtWidgets

import design

import numpy as np

import rpy2.robjects as robjects

#Класс UI приложения

class pyApp(QtWidgets.QMainWindow, design.Ui\_MainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

#Получение списка индексов округа и приведение его к нужному типу

list\_r = robjects.r(f"fun.zip()")

list\_r = np.asarray(list\_r)

list\_r = np.array(map(str, list\_r))

list\_r = list\_r.tolist()

list\_r = list(list\_r)

list\_r.sort()

self.list\_r = list\_r

self.listZip.addItems(self.list\_r) #Передача индексов в интерфейс для возможности дальнейшего выбора

self.listZip.setCurrentRow(0)

self.pushButtonPredict.clicked.connect(self.Predict) #Присоединение к кнопке функции

#Функция возвращает предсказанное значение

def Predict(self):

#Получение параметров из формы интерфейса

sqftLiving = self.spinBoxSqftLiving.value()

yrBuilt = self.spinBoxYrBuilt.value()

grade = self.spinBoxGrade.value()

zipcode = self.listZip.currentRow()

zipcode = self.list\_r[zipcode]

if self.radioButtonYes.isChecked():

waterfront = 1

elif self.radioButtonNo.isChecked():

waterfront = 0

#Получение предсказанного значения

predictFromR = robjects.r(f"fun.predict({sqftLiving}, {yrBuilt}, {grade}, {waterfront}, {zipcode})")

predictFromR = int(predictFromR[0])

#Проверка предсказанного числа на корректность и его вывод экран

if predictFromR < 0:

#В случае ошибки возникает окно с сообщением об ошибке

self.lcdNumberResult.setProperty("intValue", 0)

msgBox = QtWidgets.QMessageBox()

msgBox.setText('Ошибка предсказания')

msgBox.setWindowTitle('Ошибка')

msgBox.exec()

else:

self.lcdNumberResult.setProperty("intValue", predictFromR)

#Инициализация R и UI

def main():

robjects.r("source('R.R')")

app = QtWidgets.QApplication(sys.argv)

window = pyApp()

window.show()

app.exec\_()

#Точка старта

if \_\_name\_\_ == '\_\_main\_\_':

main()

Файл design.py:

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

# Form implementation generated from reading ui file 'design.ui'

#

# Created by: PyQt5 UI code generator 5.11.3

#

# WARNING! All changes made in this file will be lost!

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui\_MainWindow(object):

def setupUi(self, MainWindow):

MainWindow.setObjectName("MainWindow")

MainWindow.setEnabled(True)

MainWindow.resize(1071, 456)

MainWindow.setMinimumSize(QtCore.QSize(1071, 456))

MainWindow.setMaximumSize(QtCore.QSize(1071, 456))

font = QtGui.QFont()

font.setPointSize(12)

MainWindow.setFont(font)

MainWindow.setToolButtonStyle(QtCore.Qt.ToolButtonIconOnly)

self.centralwidget = QtWidgets.QWidget(MainWindow)

self.centralwidget.setMinimumSize(QtCore.QSize(1071, 430))

self.centralwidget.setMaximumSize(QtCore.QSize(1071, 430))

self.centralwidget.setObjectName("centralwidget")

self.gridLayoutWidget\_2 = QtWidgets.QWidget(self.centralwidget)

self.gridLayoutWidget\_2.setGeometry(QtCore.QRect(30, 30, 1011, 231))

self.gridLayoutWidget\_2.setObjectName("gridLayoutWidget\_2")

self.horizontalLayout\_2 = QtWidgets.QHBoxLayout(self.gridLayoutWidget\_2)

self.horizontalLayout\_2.setSizeConstraint(QtWidgets.QLayout.SetDefaultConstraint)

self.horizontalLayout\_2.setContentsMargins(0, 0, 0, 0)

self.horizontalLayout\_2.setObjectName("horizontalLayout\_2")

spacerItem = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Fixed, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_2.addItem(spacerItem)

self.gridLayout\_4 = QtWidgets.QGridLayout()

self.gridLayout\_4.setSizeConstraint(QtWidgets.QLayout.SetDefaultConstraint)

self.gridLayout\_4.setObjectName("gridLayout\_4")

self.spinBoxSqftLiving = QtWidgets.QSpinBox(self.gridLayoutWidget\_2)

self.spinBoxSqftLiving.setMinimum(200)

self.spinBoxSqftLiving.setMaximum(8000)

self.spinBoxSqftLiving.setSingleStep(50)

self.spinBoxSqftLiving.setProperty("value", 1000)

self.spinBoxSqftLiving.setObjectName("spinBoxSqftLiving")

self.gridLayout\_4.addWidget(self.spinBoxSqftLiving, 1, 1, 1, 1)

self.spinBoxGrade = QtWidgets.QSpinBox(self.gridLayoutWidget\_2)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Fixed)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.spinBoxGrade.sizePolicy().hasHeightForWidth())

self.spinBoxGrade.setSizePolicy(sizePolicy)

self.spinBoxGrade.setMinimum(4)

self.spinBoxGrade.setMaximum(12)

self.spinBoxGrade.setProperty("value", 5)

self.spinBoxGrade.setObjectName("spinBoxGrade")

self.gridLayout\_4.addWidget(self.spinBoxGrade, 3, 1, 1, 1)

self.label = QtWidgets.QLabel(self.gridLayoutWidget\_2)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Minimum)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.label.sizePolicy().hasHeightForWidth())

self.label.setSizePolicy(sizePolicy)

self.label.setObjectName("label")

self.gridLayout\_4.addWidget(self.label, 1, 0, 1, 1)

self.label\_3 = QtWidgets.QLabel(self.gridLayoutWidget\_2)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Minimum)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.label\_3.sizePolicy().hasHeightForWidth())

self.label\_3.setSizePolicy(sizePolicy)

self.label\_3.setObjectName("label\_3")

self.gridLayout\_4.addWidget(self.label\_3, 3, 0, 1, 1)

self.label\_2 = QtWidgets.QLabel(self.gridLayoutWidget\_2)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Maximum, QtWidgets.QSizePolicy.Minimum)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.label\_2.sizePolicy().hasHeightForWidth())

self.label\_2.setSizePolicy(sizePolicy)

self.label\_2.setObjectName("label\_2")

self.gridLayout\_4.addWidget(self.label\_2, 2, 0, 1, 1)

self.spinBoxYrBuilt = QtWidgets.QSpinBox(self.gridLayoutWidget\_2)

self.spinBoxYrBuilt.setMinimum(1900)

self.spinBoxYrBuilt.setMaximum(2015)

self.spinBoxYrBuilt.setProperty("value", 1950)

self.spinBoxYrBuilt.setObjectName("spinBoxYrBuilt")

self.gridLayout\_4.addWidget(self.spinBoxYrBuilt, 2, 1, 1, 1)

self.groupBox = QtWidgets.QGroupBox(self.gridLayoutWidget\_2)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed, QtWidgets.QSizePolicy.Fixed)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.groupBox.sizePolicy().hasHeightForWidth())

self.groupBox.setSizePolicy(sizePolicy)

self.groupBox.setMinimumSize(QtCore.QSize(0, 70))

self.groupBox.setFlat(True)

self.groupBox.setCheckable(False)

self.groupBox.setObjectName("groupBox")

self.verticalLayoutWidget = QtWidgets.QWidget(self.groupBox)

self.verticalLayoutWidget.setGeometry(QtCore.QRect(10, 30, 281, 30))

self.verticalLayoutWidget.setObjectName("verticalLayoutWidget")

self.horizontalLayout\_3 = QtWidgets.QHBoxLayout(self.verticalLayoutWidget)

self.horizontalLayout\_3.setContentsMargins(0, 0, 0, 0)

self.horizontalLayout\_3.setObjectName("horizontalLayout\_3")

self.radioButtonNo = QtWidgets.QRadioButton(self.verticalLayoutWidget)

self.radioButtonNo.setChecked(True)

self.radioButtonNo.setObjectName("radioButtonNo")

self.horizontalLayout\_3.addWidget(self.radioButtonNo)

self.radioButtonYes = QtWidgets.QRadioButton(self.verticalLayoutWidget)

self.radioButtonYes.setObjectName("radioButtonYes")

self.horizontalLayout\_3.addWidget(self.radioButtonYes)

self.gridLayout\_4.addWidget(self.groupBox, 0, 0, 1, 2)

self.horizontalLayout\_2.addLayout(self.gridLayout\_4)

spacerItem1 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Fixed, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_2.addItem(spacerItem1)

self.verticalLayout\_2 = QtWidgets.QVBoxLayout()

self.verticalLayout\_2.setSizeConstraint(QtWidgets.QLayout.SetNoConstraint)

self.verticalLayout\_2.setObjectName("verticalLayout\_2")

spacerItem2 = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Expanding)

self.verticalLayout\_2.addItem(spacerItem2)

self.label\_4 = QtWidgets.QLabel(self.gridLayoutWidget\_2)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Maximum, QtWidgets.QSizePolicy.Preferred)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.label\_4.sizePolicy().hasHeightForWidth())

self.label\_4.setSizePolicy(sizePolicy)

self.label\_4.setObjectName("label\_4")

self.verticalLayout\_2.addWidget(self.label\_4)

self.listZip = QtWidgets.QListWidget(self.gridLayoutWidget\_2)

self.listZip.setEnabled(True)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Expanding)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.listZip.sizePolicy().hasHeightForWidth())

self.listZip.setSizePolicy(sizePolicy)

font = QtGui.QFont()

font.setPointSize(12)

self.listZip.setFont(font)

self.listZip.setObjectName("listZip")

self.verticalLayout\_2.addWidget(self.listZip)

spacerItem3 = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Expanding)

self.verticalLayout\_2.addItem(spacerItem3)

self.horizontalLayout\_2.addLayout(self.verticalLayout\_2)

spacerItem4 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Fixed, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_2.addItem(spacerItem4)

self.horizontalLayoutWidget = QtWidgets.QWidget(self.centralwidget)

self.horizontalLayoutWidget.setGeometry(QtCore.QRect(30, 280, 1011, 131))

self.horizontalLayoutWidget.setObjectName("horizontalLayoutWidget")

self.horizontalLayout = QtWidgets.QHBoxLayout(self.horizontalLayoutWidget)

self.horizontalLayout.setContentsMargins(0, 0, 0, 0)

self.horizontalLayout.setObjectName("horizontalLayout")

self.pushButtonPredict = QtWidgets.QPushButton(self.horizontalLayoutWidget)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Minimum)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.pushButtonPredict.sizePolicy().hasHeightForWidth())

self.pushButtonPredict.setSizePolicy(sizePolicy)

font = QtGui.QFont()

font.setPointSize(24)

self.pushButtonPredict.setFont(font)

self.pushButtonPredict.setObjectName("pushButtonPredict")

self.horizontalLayout.addWidget(self.pushButtonPredict)

spacerItem5 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Fixed, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout.addItem(spacerItem5)

self.lcdNumberResult = QtWidgets.QLCDNumber(self.horizontalLayoutWidget)

self.lcdNumberResult.setDigitCount(10)

self.lcdNumberResult.setMode(QtWidgets.QLCDNumber.Dec)

self.lcdNumberResult.setProperty("intValue", 1225000)

self.lcdNumberResult.setObjectName("lcdNumberResult")

self.horizontalLayout.addWidget(self.lcdNumberResult)

MainWindow.setCentralWidget(self.centralwidget)

self.menubar = QtWidgets.QMenuBar(MainWindow)

self.menubar.setGeometry(QtCore.QRect(0, 0, 1071, 26))

self.menubar.setObjectName("menubar")

MainWindow.setMenuBar(self.menubar)

self.retranslateUi(MainWindow)

self.listZip.setCurrentRow(-1)

QtCore.QMetaObject.connectSlotsByName(MainWindow)

MainWindow.setTabOrder(self.pushButtonPredict, self.spinBoxSqftLiving)

MainWindow.setTabOrder(self.spinBoxSqftLiving, self.spinBoxYrBuilt)

MainWindow.setTabOrder(self.spinBoxYrBuilt, self.spinBoxGrade)

MainWindow.setTabOrder(self.spinBoxGrade, self.listZip)

MainWindow.setTabOrder(self.listZip, self.radioButtonNo)

MainWindow.setTabOrder(self.radioButtonNo, self.radioButtonYes)

def retranslateUi(self, MainWindow):

\_translate = QtCore.QCoreApplication.translate

MainWindow.setWindowTitle(\_translate("MainWindow", "Предсказать стоимость дома"))

self.label.setText(\_translate("MainWindow", "Жилая площадь"))

self.label\_3.setText(\_translate("MainWindow", "Оценка"))

self.label\_2.setText(\_translate("MainWindow", "Год постройки"))

self.groupBox.setTitle(\_translate("MainWindow", "Расположение на набережной:"))

self.radioButtonNo.setText(\_translate("MainWindow", "Нет"))

self.radioButtonYes.setText(\_translate("MainWindow", "Да"))

self.label\_4.setText(\_translate("MainWindow", "Индекс:"))

self.listZip.setSortingEnabled(True)

self.pushButtonPredict.setText(\_translate("MainWindow", "Предсказать"))