МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование кафедры)

01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки)

Компьютерные технологии и математическое моделирование
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка интеллектуальной системы для определения справедливых цен на жилую недвижимость»

Обучающийся В.С. Автушенко
(Инициалы Фамилия) (личная подпись)

Руководитель к.т.н, В.С. Климов
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2023

Аннотация

Тема бакалаврской работы – «Разработка интеллектуальной системы для определения справедливых цен на жилую недвижимость». Актуальность исследования заключается в развитии способов применения технологий искусственного интеллекта для автоматизации решения рутинных задач человека. В данном исследовании автоматизируется процесс определения справедливых цен квартир. Интеллектуальная система, решающая данную задачу, может быть полезна продавцам, покупателям и брокерам недвижимости, а также банкам, выдающим ипотечные кредиты.

Объектом исследования бакалаврской работы является определение справедливых цен на жилую недвижимость. Предметом исследования бакалаврской работы является интеллектуальная система и алгоритм для определения справедливых цен на недвижимость.

Цель выпускной квалификационной работы – разработка интеллектуальной системы для определения справедливых цен на жилую недвижимость. Задачами исследования является:

– анализ проблем определения справедливых цен на недвижимость;

– разработка алгоритма определения цен на квартиры;

– программная реализация интеллектуальной системы и ее тестирование.

Данная работа состоит из введения, трех глав, заключения и списка используемой литературы. В первой главе работы описываются существующие проблемы определения справедливых цен на недвижимость и анализируются пути их решения, во второй главе приводится разработанный алгоритм, в третьей главе описана практическая реализация интеллектуальной системы.

Бакалаврская работа состоит из 50 страниц текста, 31 рисунка, 1 таблицы и 21 источника.

Abstract

The topic of the bachelor's thesis is "Development of intelligent system for determining the fair prices of residential real estate". The relevance of the research is to develop ways of applying artificial intelligence technologies to automate the solution of human routine tasks. This study automates the process of determining fair prices of apartments. An intelligent system that solves this problem can be useful to sellers, buyers and brokers of real estate, as well as banks that issue mortgages.

The object of the bachelor's thesis research is the determination of fair prices of residential real estate. The subject of the bachelor's thesis is an intelligent system and algorithm for determining the fair prices of real estate.

The purpose of the graduate qualification work is to develop an intelligent system for determining fair prices for residential real estate. The objectives of the research are:

- analysis of the problems of determining fair prices for real estate;
- development of an algorithm for determining fair prices;
- software implementation of the intelligent system and its testing.

This work consists of an introduction, three chapters, a conclusion and a list of references.

The first chapter of the work describes the existing problems of determining the fair prices for real estate and analyzes the ways to solve them, the second chapter presents the developed algorithm, the third chapter describes the practical implementation of the intelligent system.

The bachelor's work consists of 50 pages of text, 31 figures, 1 table and 21 sources.

Оглавление

Введение

Оценка стоимости недвижимости применяется как организациями при реализации своей коммерческой деятельности, так и физическими лицами планирующих покупку или продажу жилья. Например, банки проводят оценку квартиры перед выдачей своему клиенту ипотечного кредита на ее покупку, чтобы понять за какую сумму возможна ее реализация, если клиент не справится с оплатой кредита. В рамках судебных заседаний в рамках гражданских процессов также может потребоваться оценка стоимости квартиры в рамках рассмотрения дела о наследстве или выплате компенсаций. Физические лица, выбирающие себе квартиру для покупки, также проводят оценку квартир, чтобы определить ее справедливую цену. Брокеры недвижимости для формирования коммерческих предложений своим клиентам также производят оценку квартир.

Благодаря развитию технологий искусственного интеллекта стало возможным автоматизация некоторых видов деятельности человека [1]. В рамках данной бакалаврской работы предлагается автоматизировать процесс оценки недвижимости.

Цель выпускной квалификационной работы – разработка интеллектуальной системы для определения справедливых цен на жилую недвижимость.

Задачами исследования является:

– анализ проблем определения справедливых цен на недвижимость;

– разработка алгоритма определения цен на квартиры;

– программная реализация интеллектуальной системы и ее тестирование.

Данная работа состоит из введения, трех глав, заключения и списка используемой литературы. В первой главе работы описываются существующие проблемы определения справедливых цен на недвижимость и

анализируются пути их решения, во второй главе приводится разработанный алгоритм, в третьей главе описана практическая реализация интеллектуальной системы.

Методы исследования – изучение литературных источников, моделирование работы алгоритмов машинного обучения, проведение вычислительных экспериментов, тестирование программной реализации предложенного алгоритма.

Практическая значимость бакалаврской работы заключается в разработке интеллектуальной системы для определения справедливых цен на жилую недвижимость.

Данная работа состоит из введения, трех глав, заключения и списка используемой литературы.

В первой главе описываются проблемы оценки недвижимости, а также а также приводится сравнительный анализ реализаций регрессионных моделей, которые потенциально можно применить для решения задачи оценки недвижимости.

Вторая глава посвящена разработке алгоритма оценки недвижимости, основанного на совместной работе двух регрессионных моделей Random forest и XGBoost.

В третьей главе описаны особенности реализации разработанного программного обеспечения, а также приведены результаты его тестирования.

В заключении коротко описываются полученные результаты исследования.

Бакалаврская работа состоит из 50 страниц текста, 31 рисунка, 1 таблицы и 21 источника.

Глава 1 Анализ проблематики определения справедливых цен на недвижимость

1.1 Обзор особенностей оценки недвижимости

Оценка стоимости недвижимости – это процесс определения справедливой рыночной цены объекта недвижимости с учетом его характеристик и текущей рыночной ситуации.

Оценка стоимости недвижимости применяется как организациями при реализации своей коммерческой деятельности, так и физическими лицами планирующих покупку или продажу жилья. Например, банки проводят оценку квартиры перед выдачей своему клиенту ипотечного кредита на ее покупку, чтобы понять за какую сумму возможна ее реализация, если клиент не справится с оплатой кредита. В рамках судебных заседаний в рамках гражданских процессов также может потребоваться оценка стоимости квартиры в рамках рассмотрения дела о наследстве или выплате компенсаций. Физические лица, выбирающие себе квартиру для покупки, также проводят оценку квартир, чтобы определить ее справедливую цену. Брокеры недвижимости для формирования коммерческих предложений своим клиентам также производят оценку квартир.

Недостатками “ручной” оценки недвижимости является:

- низкая производительность;
- наличие человеческого фактора, выраженного учете своих интересов и, как следствие, завышение и занижение оценочной стоимости квартиры;
- отсутствие учета текущих рыночных цен на надвижимость.

Поэтому перспективным направлением исследования является разработка алгоритма автоматизированной оценки недвижимости на основе модели, полученной с помощь машинного обучения.

1.2 Обзор моделей для описания зависимостей между входными параметрами и выходными

В работе Невзорова В. А. указано, что для описания зависимостей между входными параметрами и выходными применятся классификационные модели, если целевое значение является дискретным и регрессионные модели, если целевое значение является непрерывным [5].

При оценке стоимости недвижимости целевым параметром является цена квартиры, которая может принимать любое значение в заданном диапазоне. Поэтому для описания зависимостей между параметрами квартир и их ценами необходимо использовать регрессионную модель.

Рассмотрим виды регрессионных моделей, используемых в машинном обучении.

Одним из видов моделей, способных описывать зависимости между входными переменными x_1, x_2, …, x_n и выходной переменной y являются нейронные сети [4]. Нейросеть состоит из нейронов, способных выполнять два основным математических действия:

– рассчитывать взвешенную сумму сигналов, поступающих на его вход;

– применять к взвешенной сумме функцию активации для расчета выходного значения.

Для настройки работы нейронной сети используется обучающая выборка, представляющая собой примеры входных значений и соответствующие им выходные значения.

Обучение нейронной сети заключается в том, чтобы подобрать такие весовые параметры w_1, w_2, …, w_n при которых она будет правильно работать на примерах из обучающей выборке данных (рисунок 1).

Основной проблемой, связанной с применением нейронных сетей, является то, что процесс их обучения носит стохастический характер. Это

означает, что даже при использовании одних и тех же данных результат обучения нейронной сети будет уникальным. В этом случае затруднительно обеспечить повторяемость результатов обучения.

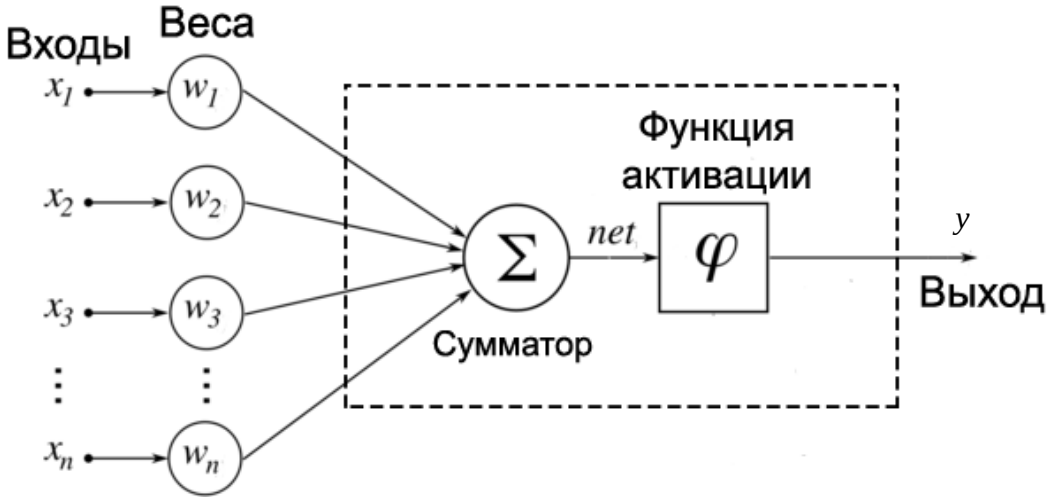

Рисунок 1 – Пример нейронной сети

В машинном обучении существуют подходы решения задач регрессионного анализа, в которых обучающая выборка изначально рассматривается как готовая регрессионная модель. Примером является алгоритм kNN. В ходе своей работы алгоритм с использованием заданной метрики оценивает схожесть исследуемого объекта с объектами из обучающей выборки.

Управляющим параметром в kNN является значение k, которое задает количество объектов из обучающей выборки, ближайших к исследуемому объекту, на основе которых будет определяться целевое выходное значение.

Если предположить, что в обучающей выборке содержится 10 объектов и каждый объект содержит признаки x_1 и x_2, то эти данные можно визуализировать в виде синих точек на декартовой системе координат (рисунок 2). Числа на графике показывают значение целевого параметра для

каждого объекта. Красной точкой на графике показан исследуемый объект, для которого необходимо определить целевое значение.

В соответствии с алгоритмом kNN для исследуемого объекта будут определены k наиболее похожих объектов из обучающей выборки. При k=3 это будут объекты, соединенные с исследуемым объектом отрезками (рисунок 1). Целевое значение исследуемого объекта в самом простом случае будет определено как среднеарифметическое целевых значений этих трех ближайших объектов из обучающей выборки.

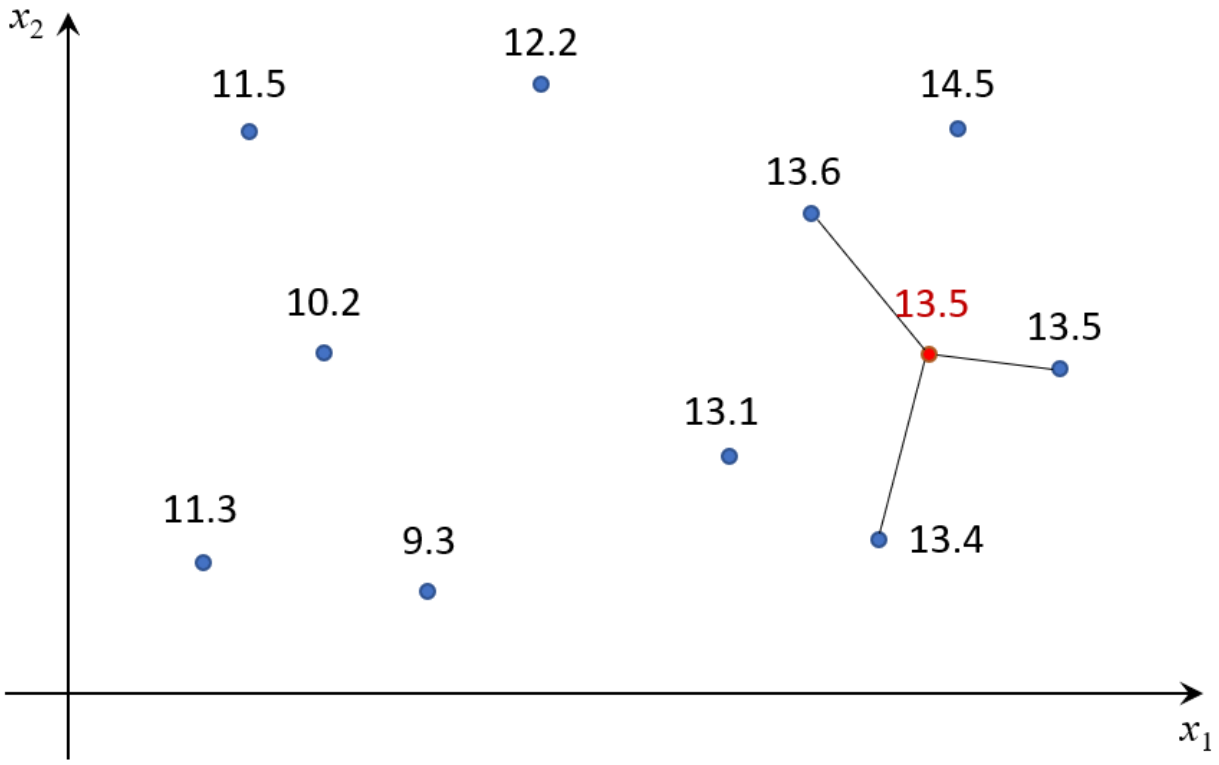

Рисунок 2 – Использования kNN для определения целевого значения *y* у объекта, отмеченного красным цветом

Другим видом моделей являются регрессионные деревья принятия решений [18].

При построении регрессионного дерева алгоритм машинного обучения условия, располагаемые в каждом узле.

Пример регрессионного дерева для обучающей выборки объектами, содержащих 2 входных признака x1 и x2 показан на рисунке 3. Преимуществами деревьев принятия решений являются:

– процесс построения дерева не носит стохастический характер, что обеспечивает повторяемость результатов обучения;

– деревья принятия решений обладают хорошей интерпретируемостью результатов работы.

-

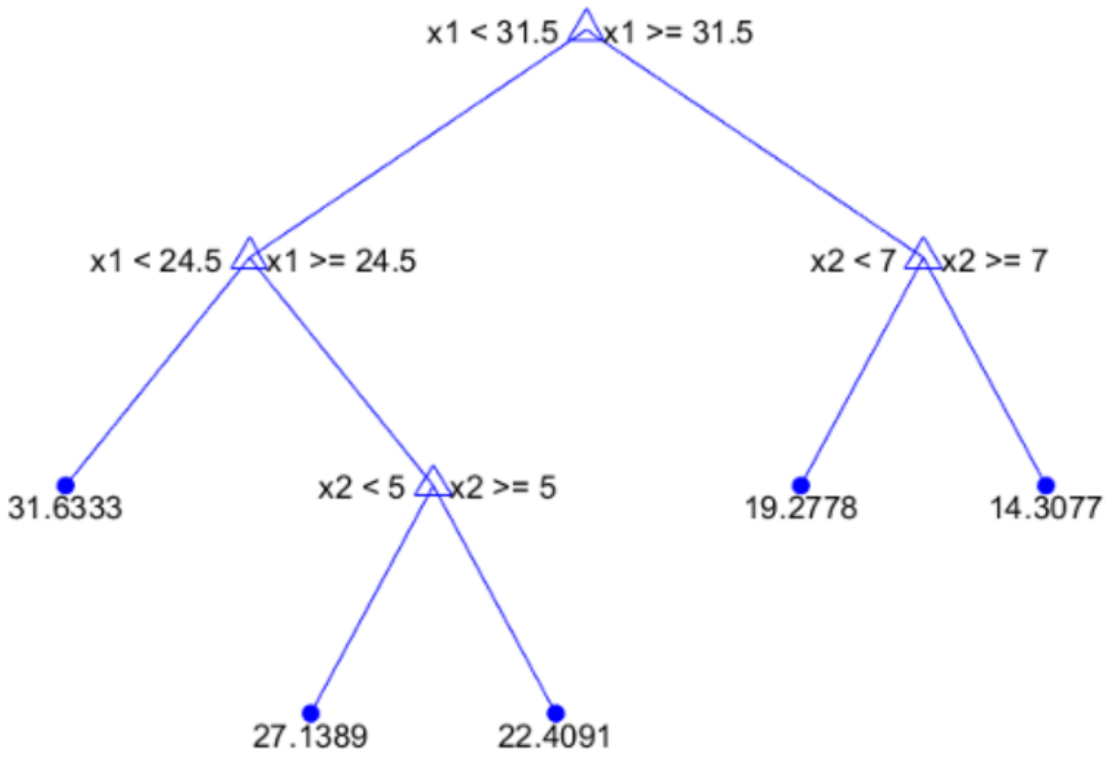

Рисунок 3 – Пример дерева регрессии

Существуют также ансамблевые регрессионные модели. Их основная идея заключается в объединении нескольких однотипных регрессионных моделей в группы для совместной работы при определении целевого значения. Наиболее известными ансамблевыми моделями являются Random forest (случайный лес) и XGBoost (градиентный бустинг) [9], [11].

Сравнение различных регрессионных моделей, реализуемых с помощью свободно распространяемых представлено в таблице 1.

Таблица 1 – Сравнение реализаций регрессионных моделей на языке программирования Python

Название	**Библиотека**	**Реализуемая технология**	**Ансамблевая модель**
MLPRegressor	scikit-learn	Многослойный персептрон	Нет
KNeighborsRegressor	scikit-learn	k – ближайших соседей	Нет
DecisionTreeRegressor	scikit-learn	Регрессионное дерево принятия решений	Нет
SVR	scikit-learn	Support Vector Regression	Нет
RandomForestRegressor	scikit-learn	Случайный лес (ансамбль деревьев)	Да
XGBoostRegressor	XGBoost	Градиентный бустинг	Да

Как отмечается во научной литературе, ансамблевые модели имеют ряд преимуществ перед одиночными [10], [13]:

– в большинстве случаев ансамблевые регрессионные модели оказываются точнее одиночных моделей;

– точность ансамблевых моделей меньше зависит от выбранных параметров алгоритма их построения, т.к. любые расхождения компенсируются большим числом составных элементов ансамбля.

Таким образом, на основе сравнения реализаций регрессионных моделей можно сделать вывод, что наиболее перспективными для решения задачи

оценки недвижимости являются ансамблевые модели Random forest и XGBoost.

Выводы по главе 1

Приведем полученные в первой главе выводы:

– определение стоимости квартиры используется: банками для снижения своих рисков при принятии решения о выдаче на нее ипотеки, брокерами недвижимости для формирования коммерческих предложений своим клиентам, покупателями недвижимости для определения справедливой стоимости жилья, судебными органами при решении вопросов наследования и определении размеров компенсации;

– процесс определения стоимости квартиры можно автоматизировать за счет применения алгоритмов машинного обучения. Для этого необходимо обучить регрессионную модель, которая на вход будет получать параметры квартиры, а на выходе будет выдавать ее прогнозную стоимость;

– проведен сравнительный анализ регрессионных моделей, реализуемых с помощью свободно распространяемых библиотек, результаты сравнения представлены в таблице 1.

Глава 2 Разработка алгоритма для оценки недвижимости на основе машинного обучения

2.1 Оценка недвижимости как задача регрессионного анализа данных

Регрессионная модель $f(A_1, A_2, A_3, \ldots, A_k, w_1, w_2, w_3, \ldots, w_n)$ в общем случае представляет собой параметрическое семейство функций, задающее отображение вида f: W×X → Y. В данном случае, W = $(w_1, w_2, w_3, \ldots, w_n)$ – параметры регрессионной модели, X = $(A_1, A_2, A_3, \ldots, A_k)$ – вектор признаков объектов, а Y – целевое, прогнозируемое моделью значение (рисунок 4).

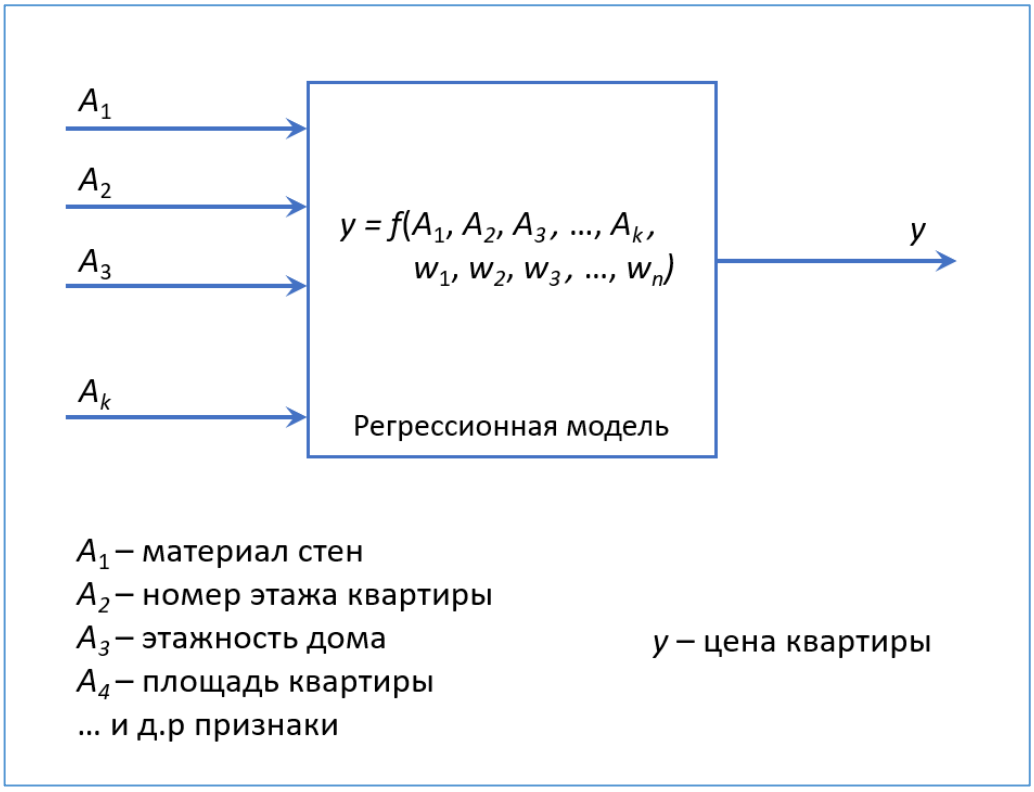

Рисунок 4 – Оценка недвижимости как задача построения регрессионной модели

Применительно к рассматриваемой в рамках данного исследования предметной области, можно представить X = (A_1, A_2, A_3, …, A_k), как набор атрибутов квартир, а Y, как цена квартиры с соответствующими параметрами. При этом значения A_1, A_2, A_3, …, A_k будут являться параметрами квартир.

Настройка параметров W регрессионной модели осуществляется с помощью обучающей выборки.

Обучающая выборка содержит в себе результаты парсинга данных из объявлений о продаже квартир в городе Москва с сервиса cian.ru. Обучающая выборка представлена текстовым файлом формата csv, пример содержимого файла показан на рисунке 5.

Размер обучающей выборки – около 65 тысяч объявлений о продаже квартир.

	A	B	C	D	E	F	G	H
1	wallsMaterial,floorNumber,floorsTotal,totalArea,kitchenArea,latitude,longitude,price							
2	brick,1,5.0,18.0,3.0,55.723379,37.628577,5600000							
3	brick,1,5.0,15.0,3.0,55.72598000000001,37.671031,4650000							
4	brick,1,5.0,11.9,1.5,55.735976,37.657817,2990000							
5	brick,1,7.0,18.4,3.0,55.786698,37.595321000000006,4390000							
6	brick,2,5.0,17.6,2.0,55.76789399999999,37.66592,4890000							
7	brick,5,9.0,21.5,3.0,55.784066,37.637425,8050000							
8	brick,5,9.0,21.5,3.0,55.784066,37.637425,8050000							
9	monolithBrick,1,3.0,29.0,6.0,55.76324,37.651825,15895000							
10	brick,5,8.0,28.8,6.5,55.776472,37.598869,10500000							
11	brick,1,5.0,25.2,2.0,55.735976,37.657817,4850000							
12	monolithBrick,5,9.0,29.0,7.0,55.775544999999994,37.672118,8450000							
13	brick,2,9.0,29.5,5.8,55.777868999999995,37.706711999999996,8449999							
14	brick,5,8.0,28.8,7.0,55.776472,37.598869,10500000							
15	monolithBrick,12,30.0,30.0,10.0,55.75461800000001,37.524058000000004,9500000							
16	brick,4,5.0,32.6,5.0,55.73478000000001,37.653909000000006,8000000							

Рисунок 5 – Содержимое файла moscow_dataset.csv

Признаки (атрибуты) обучающей выборки представлены следующими элементами:

– материал стен (wallsMaterials) – категориальный тип данных;
– номер этажа (floorNumber) – числовой тип данных;
– этажность дома (floorsTotal) – числовой тип данных;
– площадь квартиры (totalArea) – числовой тип данных;
– площадь кухни (kitchenArea) – числовой тип данных;
– долгота (latitude) – числовой тип данных;
– широта (longitude) – числовой тип данных.

Целевым значением, которое регрессионная модель должна научиться предсказывать, является цена квартиры (price).

2.2 Ансамблевые алгоритмы машинного обучения

Ансамблевые алгоритмы машинного обучения направлены на получение каскадных моделей, состоящих из однотипных элементов. Ансамблевые модели отличаются более высокой точностью прогнозирования, но при этом для своего обучения требуют обучающие выборки большой размерности.

В нашем случае размер обучающей выборки (>65 тысяч объектов) подходит для настройки ансамблевых моделей.

Наиболее известными ансамблевыми моделями являются Random forest и XGBoost. В обоих случаях в качестве составных элементов ансамблей используются деревья принятия решений. Однако концепции формирования ансамблей у этих моделей коренным образом различаются.

Модель Random forest основана на использовании технологии бэггинга (Bootstrap aggregating) [15]. Данное технология подразумевает разделение исходного набора данных на *n* частей [16].

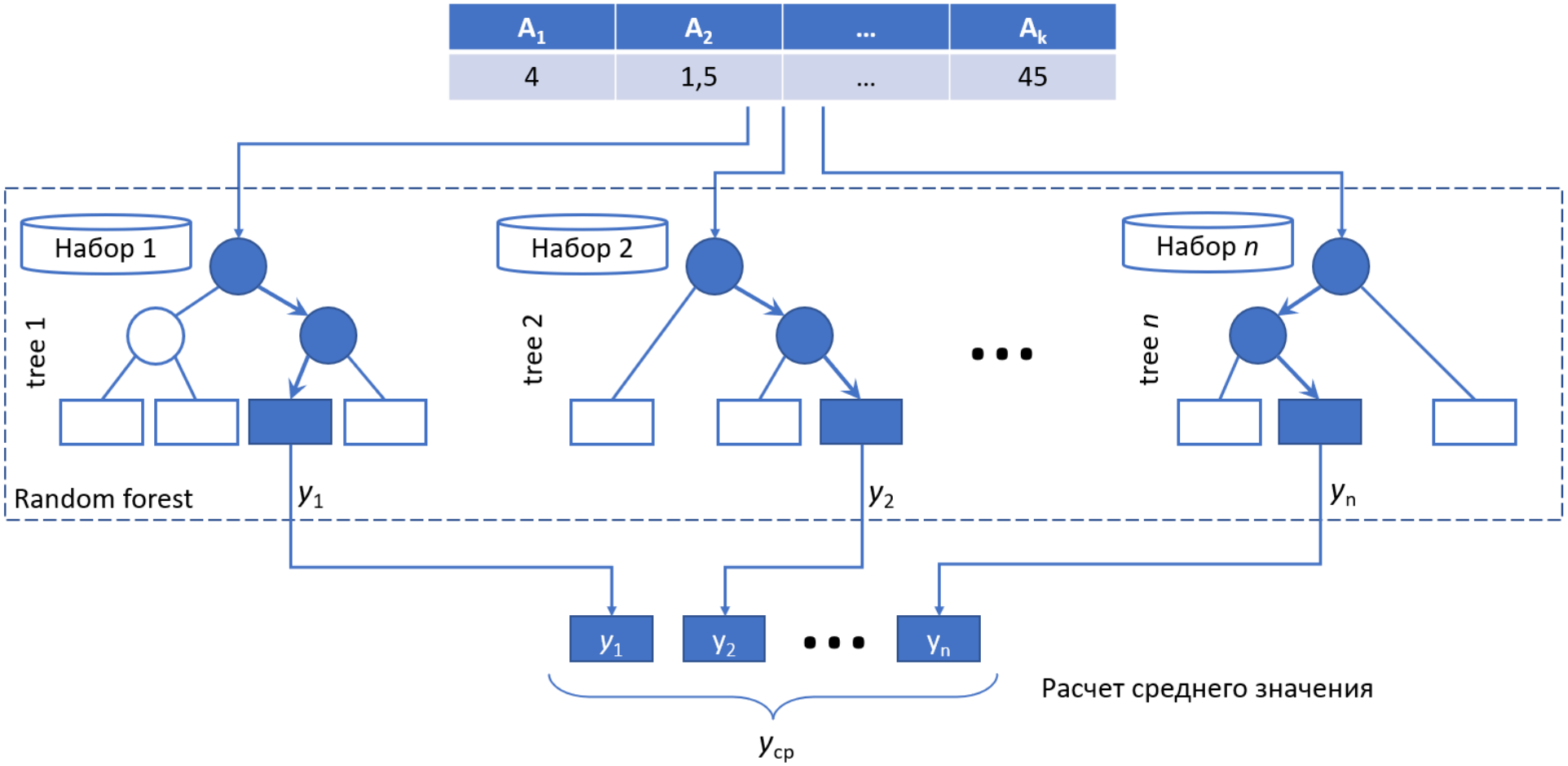

Рисунок 6 – Принцип работы ансамблевой регрессионной модели Random forest

Разделение исходного набора данных на *n* частей приводит к появлению поднабор, каждый из которых используется для независимого обучения своей регрессионной модели [17]. Набор 1 используется для получения дерева регрессии tree 1, набор 2 используется для получения дерева регрессии tree 2 и т.д.

Так как наборы данных отличаются друг от друга, то и деревья tree 1, …, tree *n* также будут различными.

В процессе работа получившегося ансамбля входные данные подаются на вход каждого имеющегося дерева и в результате чего на выходе получают набор выходных значений $y_1, y_2, \ldots, y_n$.

Для того, чтобы итоговый результат работы ансамбля деревьев решений рассчитывают среднее значение $y_{ср}$ для набора $y_1, y_2, \ldots, y_n$.

Принцип работы ансамблевой регрессионной модели Random forest показан на рисунке 6.

Регрессионная модель XGBoost основана на использовании технологии градиентного бустинга [14].

При градиентном бустинге модели ансамбля обучаются не параллельно, а последовательно. Сначала на всех имеющихся данных обучается первая модель ансамбля tree 1. Затем эта модель тестируется на тестовой выборке данных для выявления ошибок в ее работе. Зафиксированные ошибки в работе tree 1 используются для обучения следующей модели tree 2 в ансамбле. Целью обучения tree 2 является компенсация ошибок модели tree1. Затем ошибки модели tree 2 используются для обучения модели tree 3 и т.д [19].

Таким образом каждая следующая модель добавляемая в ансамбль призвана компенсировать ошибки, возникающие на предыдущих итерациях добавления моделей.

Применение градиентного бустинга для формирования ансамбля регрессионных деревьев показано на рисунке 7.

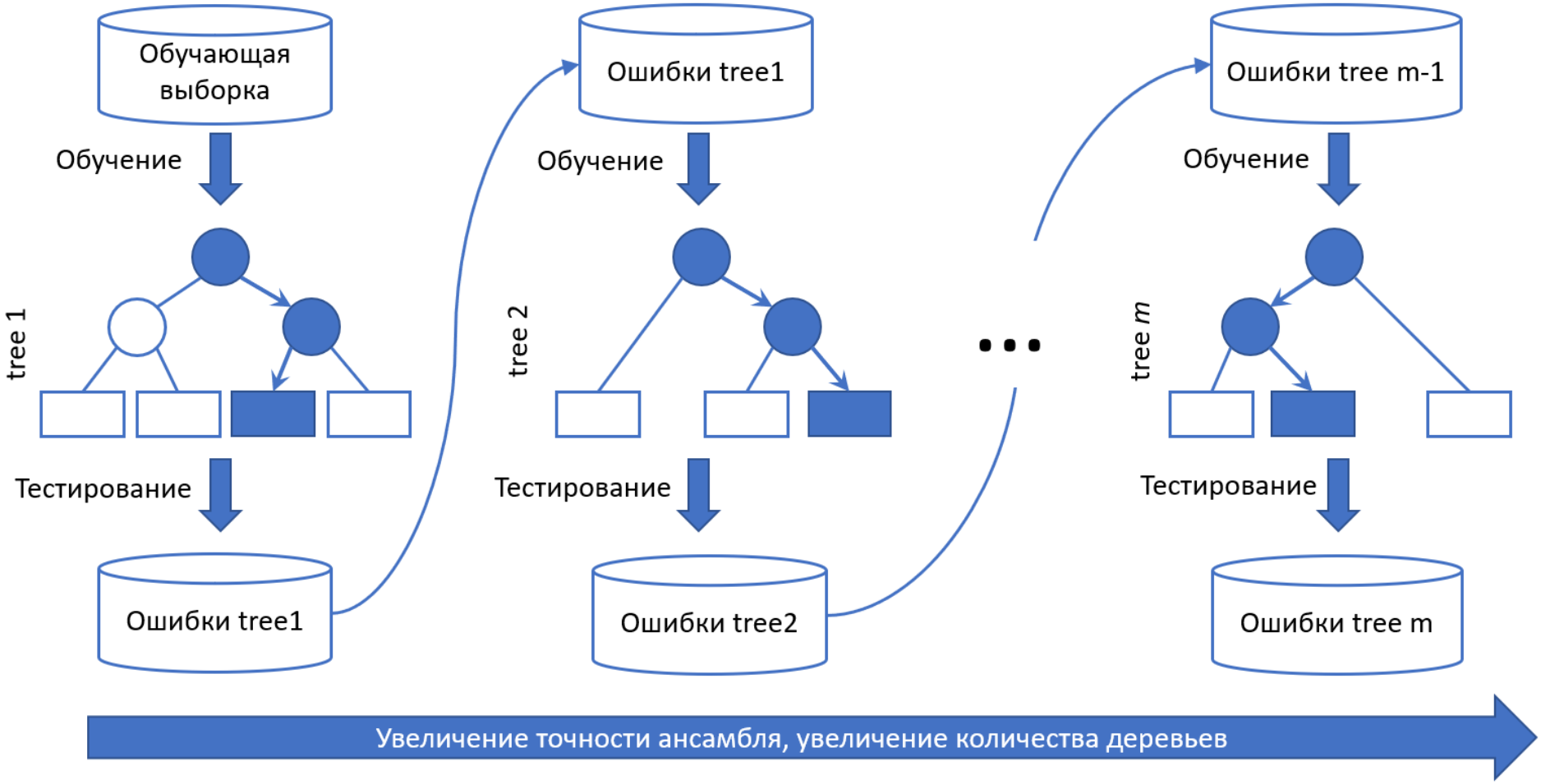

Рисунок 7 – Применение градиентного бустинга для формирования ансамбля регрессионных деревьев

Теперь необходимо обеспечить возможность согласованной работы двух регрессионных моделей, обученных независимо друг от друга.

2.3 Применение гибридного машинного обучения для увеличения точности прогнозирования цен на недвижимость

Технология согласования совместной работы двух или более разнородных регрессионных моделей носит название гибридное машинное обучение [12]. Гибридное машинное обучение отличается от ансамблевых моделей тем, что в последним случае требуется согласовать несколько однородных моделей [21].

Предложено совместную работу моделей Random forest и XGBoost реализовать следующим образом (рисунок 8). Для оценки квартиры с параметрами A_1, A_2, A_3, …, A_k производится передача этих входных данных в модели Random forest и XGBoost. Работая независимо друг от друга модель Random forest формирует на выходе оценку квартиры y_{rf}, а модель XGBoost – оценку квартиры y_{xgb}.

Теперь на основе двух полученных результатов y_{rf} и y_{xgb} формируется итоговая оценка квартиры $y(y_{rf}, y_{xgb})$. Для этого производится суммирование значений y_{rf} и y_{xgb} с разными коэффициентами.

Прогнозы от этих двух моделей смешиваются в пропорции в зависимости от коэффициента k, который принимает значение в диапазоне от 0 до 1. При k=0 учитывается только прогноз модели XGBoost, при k=1 учитывается только прогноз модели Random forest, при k=0,5 прогнозы от обеих моделей учитываются в равных пропорциях.

Таким образом, функция $y(y_{rf}, y_{xgb}) = k \cdot y_{rf},+ (k – 1) \cdot y_{xgb}$ будет выражать итоговую оценку квартиры полученную на основе двух оценок независимо работающих регрессионных моделей.

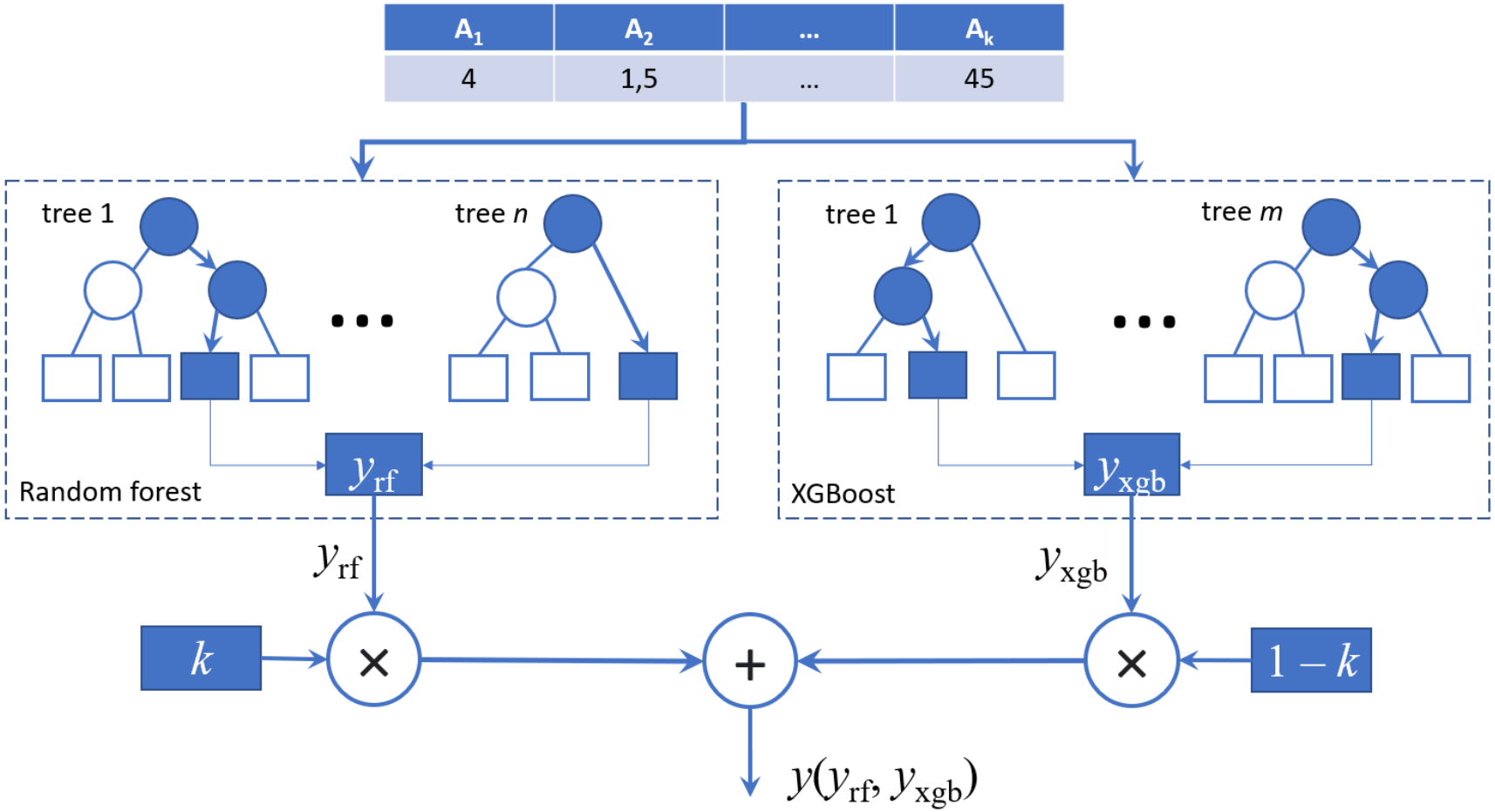

Рисунок 8 – Принцип работы гибридного машинного обучения

Теперь, когда разработан и описан принцип совместной работы двух регрессионных моделей для формирования конечной оценки квартиры перейдем к описанию алгоритма прогнозирования цен на недвижимость.

2.5 Алгоритм прогнозирования цен на недвижимость

Предложенный алгоритм автоматизированной оценки недвижимости состоит из следующих этапов:

– подготовка данных, в ходе которого осуществляется формирование обучающей выборки с примерами характеристик квартир и соответствующим им цен;

– построение регрессионных моделей, в ходе которого осуществляется независимое друг от друга обучение двух регрессоров Random forest и XGBoost;

– настройка совместной работы регрессионных моделей, в ходе которого проводится серия вычислительных экспериментов по работе совмещенной модели для определения оптимального значения коэффициента *k,* меняющегося в диапазоне от 0 до 1 с шагом 0,1.

– прогнозирование цены квартиры, в ходе которого по заданным пользователем параметрам недвижимости с использованием совмещенной модели, состоящей из двух регрессоров, проводится оценка квартиры (рисунок 9).

Этап подготовки данных включает в себя: сбор данных, кодирование категориальных признаков, очистка выборки от объектов с фиктивными значениями, расчет производных признаков (азимут, расстояние до центра города, цена за метр). Этап построения регрессионных моделей включает в себя обучение регрессора Random forest и XGBoost. Этап настройки работы регрессионных моделей включает в себя поиск оптимального значения k. Этап прогнозирования включает в себя оценку стоимости квартиры.

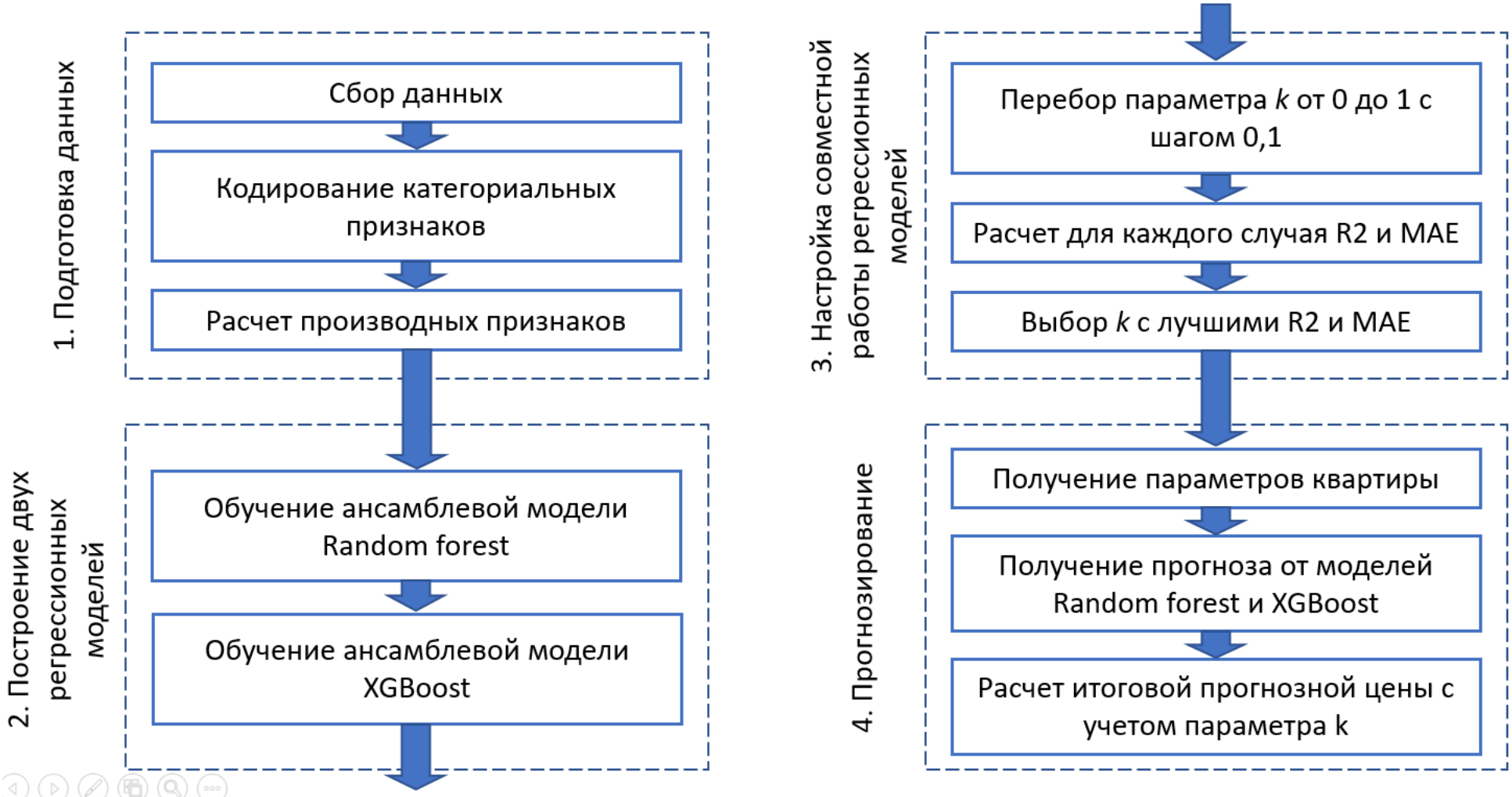

Рисунок 9 – Алгоритм прогнозирования цен на недвижимость

Теперь необходимо реализовать предложенный алгоритм в виде программного обеспечения и протестировать эффективность его работы.

Выводы по главе 2

Приведем полученные во второй главе выводы:

− предложен алгоритм оценки стоимости квартиры, который включает в себя 4 этапа: подготовка данных, которые будут использоваться в качестве примеров при обучении регрессионных моделей; независимое обучение на этих данных двух регрессионных моделей Random forest и XGBoost; настройка совместной работы этих регрессионных моделей для формирования итогового прогноза цены квартиры; использование полученной гибридной модели для прогнозирования цены квартиры с заданными пользователем параметрами недвижимости.

− этап подготовки данных включает в себя сбор данных, кодирование категориальных признаков, отсеивание записей с аномальными значениями и расчет производных признаков. Этап построения регрессионных моделей включает в себя обучение модели Random forest и обучение модели XGBoost. Этап настройки совместной работы моделей включает в себя подбор оптимального коэффициента k. Этап прогнозирования включает в себя получение от пользователя параметров квартиры и расчет итоговой цены с учетом параметра k.

Глава 3 Разработка интеллектуальной системы для оценки недвижимости

3.1 Особенности реализации системы

На языке программирования Python создано программное обеспечение, осуществляющее оценку стоимости недвижимости с использованием регрессионных моделей Random forest и XGBoost. Разработанная интеллектуальная система обладает следующим функционалом:

– загрузка данных с параметрами квартир из файла формата csv, которые в последствии используются в качестве обучающей выборки при настройке регрессионных моделей;

– расчет дополнительных производных признаков (параметров квартир), таких как расстояние до центра города и азимут, цена за квадратный метр;

– оценка точности предсказания цен на квартиры с использованием таких метрик, как средняя абсолютная процентная ошибка, медианная абсолютная процентная ошибка, коэффициент детерминации;

– кодирование категориальных признаков в данных (материал стен) в числовой формат;

– фильтрация данных с аномальными значениями (выбросами);

– обучение и моделирование работы регрессора XGBoost, основанного на градиентном бустинге;

– обучение и моделирование работы регрессора Random forest, основанного на применении ансамбля деревьев принятия решений;

– автоматизированное разделение исходных данных на обучающую и тестовую выборку;

– проведение вычислительных экспериментов по подбору долей участия регрессионных моделей XGBoost и Random forest в формировании конечной оценки стоимости квартиры;

– расчёт значимости параметров квартир при определении их влияния на конечную стоимость;

– прогнозирование цены для квартиры с произвольными параметрами, заданными пользователем.

Для снижения трудозатрат при разработке программной реализации системы оценки недвижимости применялись библиотеки различного назначения (рисунок 10).

```
import xgboost as xgb
import pandas as pd
import numpy as np
from geopy.distance import geodesic
import math
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
from sklearn.metrics import median_absolute_error
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
%matplotlib inline
%config InlineBackend.figure_format = 'retina'
import warnings
warnings.filterwarnings('ignore')
```

Рисунок 10 – Библиотеки, используемые в системе для оценки недвижимости

Приведем описание используемых библиотек:

– xgboost – библиотека, реализующая градиентный бустинг в задачах построения регрессионных моделей;

– pandas – библиотека, предоставляющая набор методов для работы с данными, которыми можно представить в табличном виде [2];

– numpy – библиотека, реализующая различные математические функции и методы по работе с массивами данных [3];

– geopy – библиотека, содержащая в себе методы для работы с географическими данными и связанными с ними системами координат;

– math – библиотека, реализующая дополнительные математические методы и функции [6];

– sklearn.metrics – раздел библиотеки sklearn, реализующий методы для расчета различных метрик, в том числе метрик для определения точности регрессионной модели [7];

– sklearn.model_selection – раздел библиотеки sklearn, реализующий методы подбора оптимальных параметров моделей регрессии и формирования обучающей и тестовой выборок данных [7];

– sklearn.ensemble – раздел библиотеки sklearn, реализующий ансамблевые модели, в том числе и Random forest;

– sklearn.preprocessing – раздел библиотеки sklearn, реализующий метод предобработки данных [8];

– matplotlib.pyplot – раздел библиотеки matplotlib, реализующий методы построения плоских графиков и диаграмм [8];

– warnings – библиотека для обработки управления предупреждениями.

В разработанной системе при оценке стоимости недвижимости учитывается расположение и удаленность квартиры относительно центра города. В исходных данных для каждой квартиры имеются географические координаты в виде пары долготы и широты, которые используются для отображения объекта на интерактивных картах. Однако для оценки стоимости квартир логичным было бы осуществить привязку каждого объекта к центру города, перейдя к системе координат «азимут», «расстояние до центра

города». Поэтому была разработана функция get_azimuth(), рассчитывающая азимут на основе долготы и широты. Программный код функции показан на рисунке 11.

```
#Вычисляет азимут по двум заданным точкам
def get_azimuth(latitude, longitude):

    rad = 6372795

    llat1 = city_center_coordinates[0]
    llong1 = city_center_coordinates[1]
    llat2 = latitude
    llong2 = longitude

    lat1 = llat1*math.pi/180.
    lat2 = llat2*math.pi/180.
    long1 = llong1*math.pi/180.
    long2 = llong2*math.pi/180.

    cl1 = math.cos(lat1)
    cl2 = math.cos(lat2)
    sl1 = math.sin(lat1)
    sl2 = math.sin(lat2)
    delta = long2 - long1
    cdelta = math.cos(delta)
    sdelta = math.sin(delta)

    y = math.sqrt(math.pow(cl2*sdelta,2)+math.pow(cl1*sl2-sl1*cl2*cdelta,2))
    x = sl1*sl2+cl1*cl2*cdelta
    ad = math.atan2(y,x)

    x = (cl1*sl2) - (sl1*cl2*cdelta)
    y = sdelta*cl2
    z = math.degrees(math.atan(-y/x))

    if (x < 0):
        z = z+180.

    z2 = (z+180.) % 360. - 180.
    z2 = - math.radians(z2)
    anglerad2 = z2 - ((2*math.pi)*math.floor((z2/(2*math.pi))) )
    angledeg = (anglerad2*180.)/math.pi

    return round(angledeg, 2)
```

Рисунок 11 – Функция расчета азимута по двум заданным точкам

Для оценки эффективности обучения регрессионных моделей в разработанном программном обеспечении применяются следующие метрики:

– средняя абсолютная процентная ошибка;

– медианная абсолютная процентная ошибка;

– коэффициент детерминации.

Расчет данных метрик осуществляется с помощью функций mean_absolute_percentage_error() и median_absolute_percentage_error(). Программный код функций показан на рисунках 12 и 13.

```
#Вычисляет среднюю абсолютную процентную ошибку
def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100
```

Рисунок 12 – Функция расчета средней абсолютной процентной ошибки

```
#Вычисляет медианную абсолютную процентную ошибку
def median_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.median(np.abs((y_true - y_pred) / y_true)) * 100
```

Рисунок 13 – Функция медианной абсолютной процентной ошибки

Для расчёта коэффициента детерминации используется стандартная функция r2_score(), реализованная в библиотеке sklearn.

Для вывода на экран рассчитанных метрик разработана функция print_metrics(). На ее вход подается два вектора значений: prediction – предсказанные регрессионной моделью значения (в нашем случае цен квартир) и val_y – фактические значения. Программный код функции показан на рисунке 14.

```
#Печатает рассчитанные значения коэффициента детерминации,
#средней и медианной абсолютных ошибок
def print_metrics(prediction, val_y):
    val_mae = mean_absolute_error(val_y, prediction)
    median_AE = median_absolute_error(val_y, prediction)
    r2 = r2_score(val_y, prediction)

    print('')
    print('R\u00b2: {:.2}'.format(r2))
    print('')
    print('Средняя абсолютная ошибка: {:.3} %'.format(mean_absolute_percentage_error(val_y
    print('Медианная абсолютная ошибка: {:.3} %'.format(median_absolute_percentage_error(v
```

Рисунок 14 – Функция вывода на экран рассчитанных метрик

Обучающая выборка данных хранится в файле moscow_dataset.csv, размещенном в облаке google drive. Для доступа к облаку google используется библиотека drive, содержащую в себе метод mount() обеспечивающий подключение к облачному хранилищу данных. Программный код для работы с облачным хранилищем данных показан на рисунке 15.

```
[ ]  from google.colab import drive
     drive.mount('/content/drive')
     path1 = "/content/drive/MyDrive/Colab Notebooks/moscow_dataset.csv"

     Drive already mounted at /content/drive; to attempt to forcibly remount,
```

Рисунок 15 – Подключение облачного хранилища с данными

Загрузка обучающей выборки с параметрами квартир осуществляется из файла moscow_dataset.csv с помощью метода read_csv(), реализованного в библиотеке pandas. В качестве параметра методу read_csv() предается путь до файла. Для контроля успешности загрузки данных с помощью метода head() выводятся первые несколько объектов обучающей выборки. Программный код представлен на рисунке 16.

```
file_path = path1
df = pd.read_csv(file_path)

#Выводим 5 первых строк датафрейма
df.head(5)
```

	wallsMaterial	floorNumber	floorsTotal	totalArea	kitchenArea	latitude	longitude	price
0	brick	1	5.0	18.0	3.0	55.723379	37.628577	5600000
1	brick	1	5.0	15.0	3.0	55.725980	37.671031	4650000
2	brick	1	5.0	11.9	1.5	55.735976	37.657817	2990000
3	brick	1	7.0	18.4	3.0	55.786698	37.595321	4390000
4	brick	2	5.0	17.6	2.0	55.767894	37.665920	4890000

Рисунок 16 – Отображение загруженных данных в виде таблицы

На основе имеющихся признаков производится расчет дополнительные параметров квартир: цены за квадратный метр, расстояние до центра города и азимут (рисунок 17).

```
#Создаем новый столбец Стоимость 1 кв.м путем построчного деления стоимостей
#квартир на их общие площади
df['priceMetr'] = df['price']/df['totalArea']

#Задаем широту и долготу центра города и рассчитываем для каждой квартиры
#расстояние от центра и азимут
city_center_coordinates = [55.7522, 37.6156]
df['distance'] = list(map(lambda x, y: geodesic(city_center_coordinates,
                          [x, y]).meters, df['latitude'], df['longitude']))
df['azimuth'] = list(map(lambda x, y: get_azimuth(x, y), df['latitude'],
                         df['longitude']))

#Выбираем из датафрейма только те квартиры, которые расположены не дальше 40 км
#от центра города с панельными стенами
df = df.loc[(df['distance'] < 40000)]

#Округляем значения стоблцов Стоимости метра, расстояния и азимута
df['priceMetr'] = df['priceMetr'].round(0)
df['distance'] = df['distance'].round(0)
df['azimuth'] = df['azimuth'].round(0)
```

Рисунок 17 – Отображение загруженных данных в виде таблицы

Расчет расстояния до центра города и азимута необходимо для того, чтобы привязать стоимость квартир к их расположению относительно центра. Расчет стоимости квадратного метра и использование его в качестве целевого значения вместо суммарной стоимости квартиры несколько упрощает задачу обучения регрессионной модели.

После расчета дополнительных параметров квартир в нашем наборе данных содержится 11 признаков, представленных на рисунке 18.

```
#Выводим сводную информацию о датафрейме и его столбцах (признаках)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 63682 entries, 0 to 63944
Data columns (total 11 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   wallsMaterial  63682 non-null  object 
 1   floorNumber    63682 non-null  int64  
 2   floorsTotal    63682 non-null  float64
 3   totalArea      63682 non-null  float64
 4   kitchenArea    63682 non-null  float64
 5   latitude       63682 non-null  float64
 6   longitude      63682 non-null  float64
 7   price          63682 non-null  int64  
 8   priceMetr      63682 non-null  float64
 9   distance       63682 non-null  float64
 10  azimuth        63682 non-null  float64
dtypes: float64(8), int64(2), object(1)
memory usage: 5.8+ MB
```

Рисунок 18 – Вывод информации о загруженных данных

В разработанном программном обеспечении реализована функция очистка имеющихся данных от аномальных значений. Аномальные значения в данных появляются, когда пользователи при размещении объявлений по продаже квартир обязательные поля заполняют фиктивными значениями. Фиктивные значения мешают настройке регрессионной модели, поэтому такие объявления подвергаются удалению. Программный код для очистки данных от аномальных значений представлен на рисунке 19.

```
#Вычисляем строки со значениями-выбросами
first_quartile = df.quantile(q=0.25)
third_quartile = df.quantile(q=0.75)
IQR = third_quartile - first_quartile
outliers = df[(df > (third_quartile + 1.5 * IQR)) | (df < (first_quartile - 1.5 * IQR))].count(axis=1)
outliers.sort_values(axis=0, ascending=False, inplace=True)

#Удаляем из датафрейма 3000 строк, подходящих под критерии выбросов
outliers = outliers.head(3000)
df.drop(outliers.index, inplace=True)
```

Рисунок 19 – Отсеивание данных с аномальным значениями

Как видно из рисунка 18 один из признаков (wallsMaterial) представляет собой означения строкового типа. Используемые в программном обеспечении библиотеки, реализующие регрессионные модели Random forest и XGBoost не умеют обрабатывать данные строкового типа. Потому была написана функция осуществляющая кодирования строковых значений признаков в числовой вид. Для этого применяется кодировщик LabelEncoder(), реализованный в библиотеке sklearn. Программный код, реализующий кодирование строковых данных в числовой вид показан на рисунке 20.

```
[ ]  #Вычисляем столбцы с категорийными признаками, затем заменяем их на числа
     categorical_columns = df.columns[df.dtypes == 'object']
     labelencoder = LabelEncoder()
     for column in categorical_columns:
         df[column] = labelencoder.fit_transform(df[column])
         print(dict(enumerate(labelencoder.classes_)))
```

Рисунок 20 – Кодирование категориальных признаков в числа

Для проверки работы кодировщика LabelEncoder() на экран еще раз выводится сводная информация об имеющихся в наборе данных признаков и их типов (рисунок 21). Как видно, признак wallsMaterial теперь имеет числовой тип int64.

```
#Выводим сводную информацию о датафрейме и его столбцах (признаках), чтобы
#убедиться, что теперь они все содержат цифровые значения
df.info()
```

```
{0: 'None', 1: 'block', 2: 'brick', 3: 'monolith', 4: 'monolithBrick', 5: 'old',
<class 'pandas.core.frame.DataFrame'>
Int64Index: 60682 entries, 0 to 63943
Data columns (total 11 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   wallsMaterial  60682 non-null  int64
 1   floorNumber    60682 non-null  int64
 2   floorsTotal    60682 non-null  float64
 3   totalArea      60682 non-null  float64
 4   kitchenArea    60682 non-null  float64
 5   latitude       60682 non-null  float64
 6   longitude      60682 non-null  float64
 7   price          60682 non-null  int64
 8   priceMetr      60682 non-null  float64
 9   distance       60682 non-null  float64
 10  azimuth        60682 non-null  float64
dtypes: float64(8), int64(3)
memory usage: 5.6 MB
```

Рисунок 21 – Вывод информации о загруженных данных после кодирования категориальных признаков

Теперь, когда данные отчислены от аномальных значений и типы всех признаков приведены в числовой вид необходимо определить какой из столбцов будет является целевым.

В качестве целевого столбца выбирается цена квадратного ветка квартиры (priceMetr). В качестве входных признаков используются столбцы: материал стен (wallsMaterials), номер этажа (floorNumber), этажность дома (floorsTotal), площадь квартиры (totalArea), площадь кухни (kitchenArea), расстояние до центра города (distance), азимут (azimuth).

Целевые значения помещаются в вектор *y*, а входные признаки помещаются в матрицу *X*. Программный код формирования переменных *X* и *y* показан на рисунке 22.

```
#Назначаем целевой переменной цену 1 кв. метра, а можно и цену всей квартиры,
#тогда будет y = df['price']
y = df['priceMetr']

#Создаем список признаков, на основании которых будем строить модели
features = [
            'wallsMaterial',
            'floorNumber',
            'floorsTotal',
            'totalArea',
            'kitchenArea',
            'distance',
            'azimuth'
           ]

#Создаем датафрейм, состоящий из признаков, выбранных ранее
X = df[features]
```

Рисунок 22 – Формирование матрицы X с атрибутами квартир и вектора y с соответствующими ценами за квадратный метр

Теперь необходимо имеющиеся данные разделить на обучающую и тестовую выборки. Обучающая выборка данных будет использоваться для обучения моделей, а тестовая – для определения точности прогнозирования цены квартир. Для разделения данных на обучающую и тестовую выборки применяется метод train_test_split(), реализованный в библиотеке sklearn (рисунок 23).

```
#Проводим случайное разбиение данных на выборки для обучения (train)
#и валидации (val), по умолчанию в пропорции 0.75/0.25
train_X, val_X, train_y, val_y = train_test_split(X, y, random_state=1)
```

Рисунок 23 – Формирование тренировочной и тестовой выборок данных

На следующем этапе осуществляется независимое обучение двух регрессионных моделей на тренировочной выборке данных и тестирование их точности работы на тестовой выборке данных. Программный код для

обучения и тестирования модели Random forest представлен на рисунке 24, а программный код модели XGBoost показан на рисунке 25.

```
#Создаем регрессионную модель случайного леса
rf_model = RandomForestRegressor(n_estimators=2000, #2000
                                 n_jobs=-1,
                                 bootstrap=False,
                                 criterion='friedman_mse',
                                 max_features=3,
                                 random_state=1,
                                 max_depth=55,
                                 min_samples_split=5
                                 )

#Проводим подгонку модели на обучающей выборке
rf_model.fit(train_X, train_y)

#Вычисляем предсказанные значения цен на основе валидационной выборки
rf_prediction = rf_model.predict(val_X).round(0)

#Вычисляем и печатаем величины ошибок при сравнении известных цен квартир из
#валидационной выборки с предсказанными моделью
print_metrics(rf_prediction, val_y)
```

```
R²: 0.84

Средняя абсолютная ошибка: 7.89 %
Медианная абсолютная ошибка: 5.36 %
```

Рисунок 24 – Обучение регрессионной модели Random forest и расчет ошибки прогнозирования

Повышение точности предсказания стоимости квартир обеспечивается с помощью совместной работы двух независимо обученных регрессионных моделей. Предсказания двух регрессионных моделей суммируются с учетом долей их участия в формировании конечного прогноза по стоимости квартиры. Результат работы такой гибридной модели с равными долями участия показан на рисунке 26.

```
#Создаем регрессионную модель XGBoost
xgb_model = xgb.XGBRegressor(objective ='reg:gamma',
                             learning_rate = 0.01,
                             max_depth = 45,
                             n_estimators = 2000,
                             nthread = -1,
                             eval_metric = 'gamma-nloglik',
                             )

#Проводим подгонку модели на обучающей выборке
xgb_model.fit(train_X, train_y)

#Вычисляем предсказанные значения цен на основе валидационной выборки
xgb_prediction = xgb_model.predict(val_X).round(0)

#Вычисляем и печатаем величины ошибок при сравнении известных цен квартир из
#валидационной выборки с предсказанными моделью
print_metrics(xgb_prediction, val_y)
```

```
R²: 0.83

Средняя абсолютная ошибка: 7.73 %
Медианная абсолютная ошибка: 5.23 %
```

Рисунок 25 – Обучение регрессионной модели XGBoost и расчет ошибки прогнозирования

```
#Усредняем предсказания обоих моделей
prediction = rf_prediction * 0.5 + xgb_prediction * 0.5

#Вычисляем и печатаем величины ошибок для усредненного предсказания
print_metrics(prediction, val_y)
```

```
R²: 0.84

Средняя абсолютная ошибка: 7.62 %
Медианная абсолютная ошибка: 5.16 %
```

Рисунок 26 – Использование гибридной модели

В библиотеке scikit-learn во все реализации алгоритмов построения регрессионных моделей встроена функция расчета значимости признаков, оценивающая долю вклада каждого параметра в конечный результат.

В разработанном программном обеспечении функция оценки значимости признаков используется для оценки степени влияния каждого параметра квартир на их стоимость.

Доступ к коэффициентам значимости признаков осуществляется через поле feature_importances_. Для отображения значимости признаков в виде столбчатой диаграммы используется метод bar() библиотеки matplotlib. Полный код для оценки значимости признаков квартир и отображения результатов в виде столбчатой диаграммы показан на рисунке 27.

```
#Рассчитываем важность признаков в модели Random forest
importances = rf_model.feature_importances_
std = np.std([tree.feature_importances_ for tree in rf_model.estimators_],
             axis=0)
indices = np.argsort(importances)[::-1]

#Печатаем рейтинг признаков
print("Рейтинг важности признаков:")
for f in range(X.shape[1]):
    print("%d. %s (%f)" % (f + 1, features[indices[f]], importances[indices[f]]))

#Строим столбчатую диаграмму важности признаков
plt.figure()
plt.title("Важность признаков")
plt.bar(range(X.shape[1]), importances[indices], color="g", yerr=std[indices],
        align="center")
plt.xticks(range(X.shape[1]), indices)
plt.xlim([-1, X.shape[1]])
plt.show()
```

Рисунок 27 – Расчет значимости признаков и построение диаграммы

Полученная в результате выполнения программного кода диаграмма показана на рисунке 28.

На диаграмме приняты следующие обозначения признаков: 0 – wallsMaterial / материал стен (0,099); 1 – floorNumber / номер этажа (0,033); 2 – floorsTotal / этажность дома (0,067); 3 – totalArea / площадь квартиры (0,065); 4 – kitchenArea / площадь кухни (0,064); 5 – distance / расстояние до центра города (0,526); 6 – azimuth / азимут (0,146).

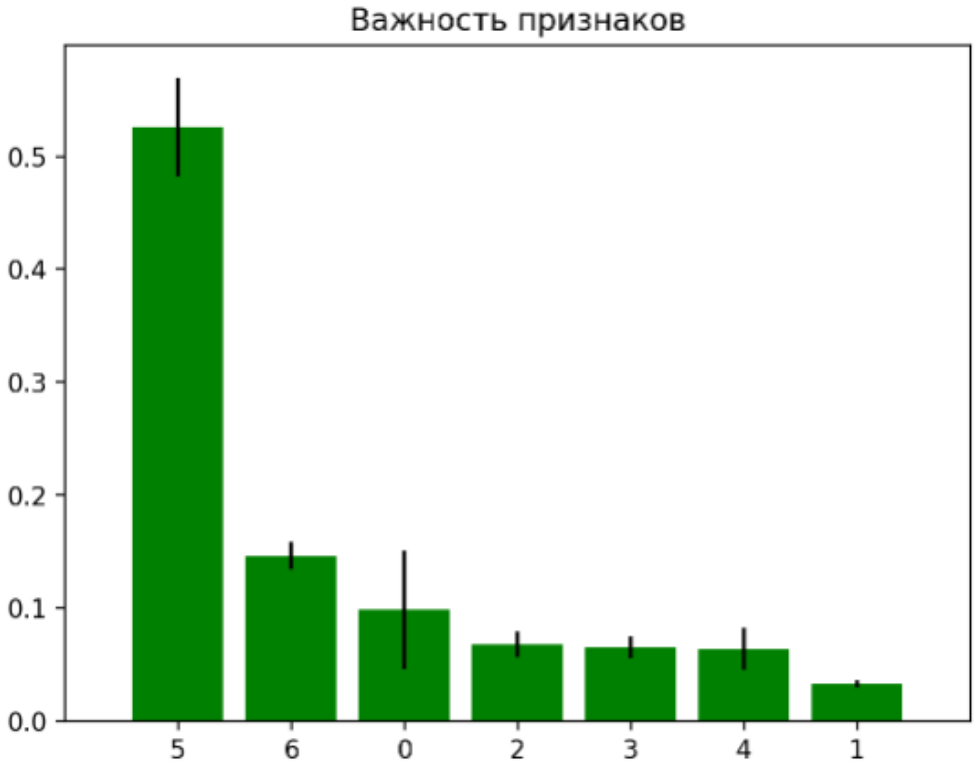

Рисунок 28 – Диаграмма значимости признаков

Как видно из диаграммы (рисунок 28) в Москве на стоимость квартиры наибольшее влияние оказывает: расстояние до центра города, азимут и материал стен.

3.2 Тестирование системы и определение оптимальных параметров ее работы

Одной из ключевых особенностей разработанного программного

обеспечения является применение технологии гибридного машинного обучения, обеспечивающей согласованную работу двух регрессионных моделей Random forest и XGBoost. Прогнозы от этих двух моделей смешиваются в пропорции в зависимости от коэффициента k, который принимает значение в диапазоне от 0 до 1. При k=0 учитывается только прогноз модели XGBoost, при k=1 учитывается только прогноз модели Random forest, при k=0,5 прогнозы от обеих моделей учитываются в равных пропорциях.

Для того, чтобы подобрать значение коэффициента k при котором обеспечивается максимальная точность совместной работы моделей проводится серия экспериментальных вычислений. При этом параметр k меняется в диапазоне от 0 до 1 с шагом 0,1 и для каждого случая рассчитывается коэффициент детерминации R2 и средняя абсолютная процентная ошибка MAE. Чем выше значение R2 и меньше значение ошибки MAE, тем точнее результаты прогнозирования. Цель проведения вычислительных экспериментов найти значение k, при котором выполняется данное условие.

Входе проведения вычислительных экспериментов формируется вектор значений k в односменной переменной, а также вектор значений R2 и вектор значений MAE. На основе полученных значений строятся с помощью методов библиотеки matplotlib строится и выводится на экран:

– график изменения коэффициента R2 в зависимости от коэффициента k (показан синим цветом);

– график изменения ошибки MAE в зависимости от коэффициента k (показан красным цветом).

Программный код для построения графиков и результат его выполнения показан на рисунке 29.

```
fig, (ax1,ax2) = plt.subplots(2, 1, sharex=True)
ax1.plot(k, R2, label= 'R2')
ax2.plot(k, mae, label = 'mae, %', color = "darkred")
ax1.legend()
ax2.legend()
plt.show()
```

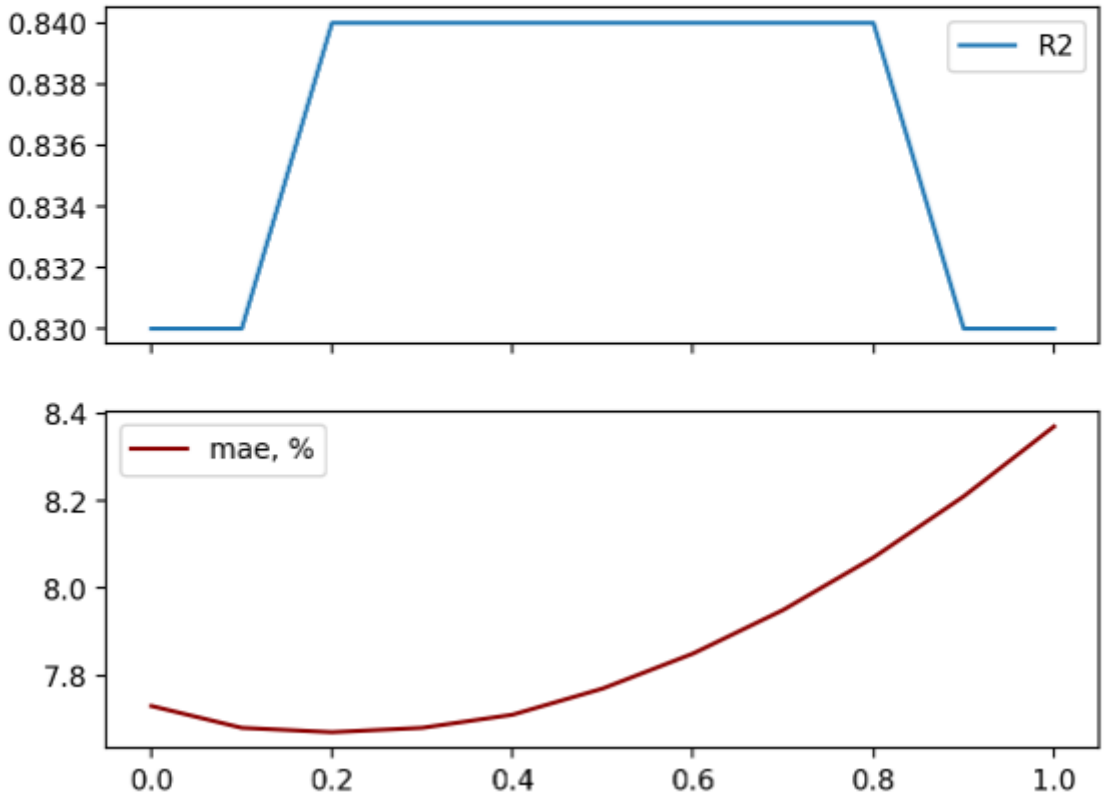

Рисунок 29 – Построение графиков зависимости коэффициента детерминации и средней абсолютной ошибки в зависимости от доли учета прогнозов моделей Random forest и XGboost при расчете конечного прогноза

Как видно из рисунка 29 оптимальным значением является k=0,2, так как при этом значении обеспечивается минимальная ошибка MAE=7,5% и максимальное значение коэффициента детерминации R2=0,84

Продемонстрируем работу программного обеспечении на примере оценки квартиры из одного отдельного объявлений из сервиса cian.ru. Текст объявления о продаже квартиры ценой 11 200 000 рублей представлен на рисунке 30.

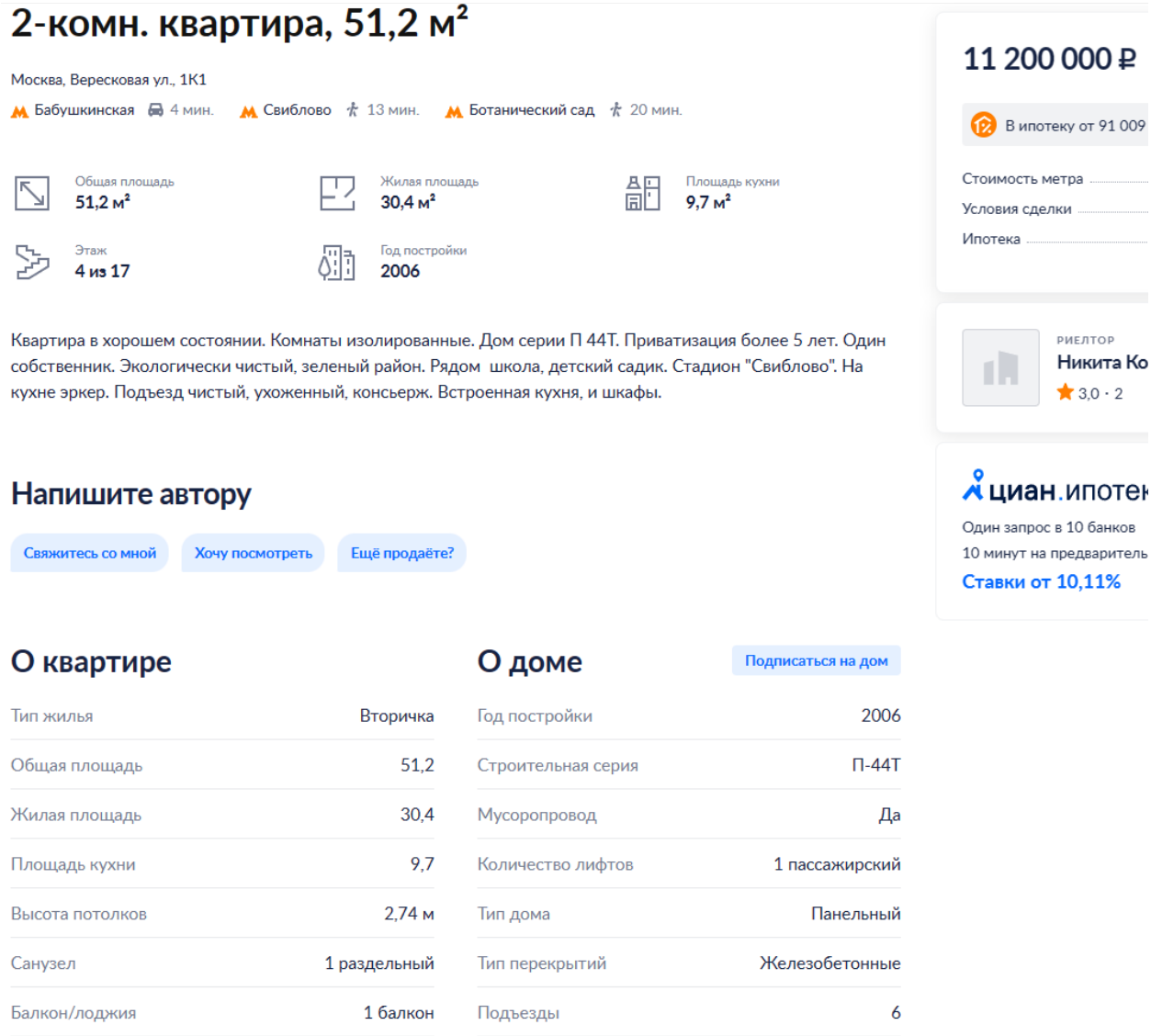

Рисунок 30 – Снимок экрана с объявлением квартиры

Предадим в разработанное программное обеспечение параметры этой квартиры и определим на сколько точно обученная модель сможет определить ее цену.

Программный код по прогнозированию цены квартиры с указанными характеристиками показан на рисунке 31.

```
flat = pd.DataFrame({
                     'wallsMaterial':[6],
                     'floorNumber':[4],
                     'floorsTotal':[17],
                     'totalArea':[51.2],
                     'kitchenArea':[9.7],
                     'latitude':[55.858817],
                     'longitude':[37.638755]
                     })

#Рассчитываем недостающие параметры квартиры - расстояние от центра города и азимут
flat['distance'] = list(map(lambda x, y: geodesic(city_center_coordinates,
                            [x, y]).meters, flat['latitude'],
                            flat['longitude']))
flat['azimuth'] = list(map(lambda x, y: get_azimuth(x, y), flat['latitude'],
                           flat['longitude']))
flat['distance'] = flat['distance'].round(0)
flat['azimuth'] = flat['azimuth'].round(0)

#Удаляем ненужные столбцы с широтой и долготой
flat = flat.drop('latitude', axis=1)
flat = flat.drop('longitude', axis=1)

#Вычисляем предсказанное значение стоимости по двум моделям
rf_prediction_flat = rf_model.predict(flat).round(0)
xgb_prediction_flat = xgb_model.predict(flat).round(0)

#Усредняем полученные знаечения и умножаем на общую площадь квартиры
price = (rf_prediction_flat * 0.5 + xgb_prediction_flat * 0.5)*flat['totalArea'][0]

#Печатаем предсказанное значение цены предложения
print(f'Предсказанная моделью цена предложения: {int(price[0].round(-3))} рублей')
```

```
Предсказанная моделью цена предложения: 11199000 рублей
```

Рисунок 31 – Задание параметров квартиры и расчет прогнозной цены

Предсказанная моделью цена квартиры составляет 11 199 000 рублей, а цена этого объекта на cian.ru составляет 11 200 000 рублей. На основе сравнения предсказанной и реальной цены можно сделать вывод, что обученная модель научилась успешно определять цены квартир в Москве. А разработанное программное обеспечение успешно решает задачу автоматизированной оценки стоимости квартир.

Выводы по главе 3

Приведем полученные в третьей главе выводы:

− с использованием языка программирования python разработано программное обеспечение для определения стоимости квартир. Стоимость квартир определяется на основе прогнозов двух ансамблевых регрессионных моделей Random forest и XGBoost;

− в разработанном программном обеспечении оценка стоимости квартир проводится на основе анализа таких признаков, как материал стен, номер этажа, этажность дома, площадь квартиры, площадь кухни, расстояние до центра города, азимут;

− для имеющегося набора данных определено оптимальное значение коэффициента k, влияющего на долю участия каждой из двух моделей в формировании конченой оценки стоимости квартиры. Оптимальным значением является k=0,2, так как при этом значении обеспечивается минимальная ошибка MAE=7,5% и максимальное значение коэффициента детерминации R2=0,84.

− тестирование разработанного программного обеспечения проводилось на объявлениях сервиса cian.ru для квартир в городе Москва. Результаты тестирования доказали состоятельность предложенных в исследовании подходов.

Заключение

При выполнении бакалаврской работы были получены следующие результаты:

− определение стоимости квартиры используется: банками для снижения своих рисков при принятии решения о выдаче на нее ипотеки, брокерами недвижимости для формирования коммерческих предложений своим клиентам, покупателями недвижимости для определения справедливой стоимости жилья, судебными органами при решении вопросов наследования и определении размеров компенсации;

− процесс определения стоимости квартиры можно автоматизировать за счет применения алгоритмов машинного обучения. Для этого необходимо обучить регрессионную модель, которая на вход будет получать параметры квартиры, а на выходе будет выдавать ее прогнозную стоимость;

− проведен сравнительный анализ регрессионных моделей, реализуемых с помощью свободно распространяемых библиотек, результаты сравнения представлены в таблице 1;

− предложен алгоритм оценки стоимости квартиры, который включает в себя 4 этапа: подготовка данных, которые будут использоваться в качестве примеров при обучении регрессионных моделей; независимое обучение на этих данных двух регрессионных моделей Random forest и XGBoost; настройка совместной работы этих регрессионных моделей для формирования итогового прогноза цены квартиры; использование полученной гибридной модели для прогнозирования цены квартиры с заданными пользователем параметрами недвижимости;

− этап подготовки данных включает в себя сбор данных, кодирование категориальных признаков, отсеивание записей с аномальными значениями и расчет производных признаков. Этап построения регрессионных моделей

включает в себя обучение модели Random forest и обучение модели XGBoost. Этап настройки совместной работы моделей включает в себя подбор оптимального коэффициента k. Этап прогнозирования включает в себя получение от пользователя параметров квартиры и расчет итоговой цены с учетом параметра k.

− с использованием языка программирования python разработано программное обеспечение для определения стоимости квартир. Стоимость квартир определяется на основе прогнозов двух ансамблевых регрессионных моделей Random forest и XGBoost;

− в разработанном программном обеспечении оценка стоимости квартир проводится на основе анализа таких признаков, как материал стен, номер этажа, этажность дома, площадь квартиры, площадь кухни, расстояние до центра города, азимут;

− для имеющегося набора данных определено оптимальное значение коэффициента k, влияющего на долю участия каждой из двух моделей в формировании конченой оценки стоимости квартиры. Оптимальным значением является k=0,2, так как при этом значении обеспечивается минимальная ошибка MAE=7,5% и максимальное значение коэффициента детерминации R2=0,84;

− тестирование разработанного программного обеспечения проводилось на объявлениях сервиса cian.ru для квартир в городе Москва. Результаты тестирования доказали состоятельность предложенных в исследовании подходов.

Список используемой литературы и используемых источников

1. Алханов А.А. Машинное обучение и его применение в современном мире // Проблемы науки. 2021. №7 (66). URL: https://cyberleninka.ru/article/n/mashinnoe-obuchenie-i-ego-primenenie-v-sovremennom-mire (дата обращения: 04.04.2023).

2. Григорьев Е.А. Разведочный анализ данных с помощью Python / Григорьев Е.А., Климов Н.С. // E-Scio. 2020. №2 (41). URL: https://cyberleninka.ru/article/n/razvedochnyy-analiz-dannyh-s-pomoschyu-python (дата обращения: 22.09.2022).

3. Гришков, Д.Ю. Язык высокого уровня программирования Python / Гришков Данила Юрьевич, Аусилова Назерке Мырзабековна // НИР/S&R. 2022. №1 (9). URL: https://cyberleninka.ru/article/n/yazyk-vysokogo-urovnya-programmirovaniya-python (дата обращения: 22.09.2022).

4. Есбаганбетов М.Б. Применение нейросетевых методов машинного обучения к задаче решения дифференциальных уравнении разных видов / Есбаганбетов Манас Байрамбаевич, Иманбаев Кайрат Советович, Тергеуов Олжас Серикович // Universum: технические науки. 2022. №5-2 (98). URL: https://cyberleninka.ru/article/n/primenenie-neyrosetevyh-metodov-mashinnogo-obucheniya-k-zadache-resheniya-differentsialnyh-uravnenii-raznyh-vidov (дата обращения: 04.04.2023).

5. Невзорова В. А. Методы машинного обучения в прогнозировании исходов и рисков сердечно-сосудистых заболеваний у пациентов с артериальной гипертензией (по материалам эссе-рфв Приморском крае) / Невзорова В. А., Плехова Н. Г., Присеко Л. Г., Черненко И. Н., Богданов Д. Ю., Мокшина М. В., Кулакова Н. В. // РКЖ. 2020. №3. URL: https://cyberleninka.ru/article/n/metody-mashinnogo-obucheniya-v-prognozirovanii-ishodov-i-riskov-serdechno-sosudistyh-zabolevaniy-u-patsientov-s-arterialnoy (дата обращения: 04.04.2023).

6. Негодин В.А. Машинное обучение в языке программирования Python // Форум молодых ученых. 2019. №8 (36). URL: https://cyberleninka.ru/article/n/mashinnoe-obuchenie-v-yazyke-programmirovaniya-python (дата обращения: 04.04.2023).

7. Разин С.А. Что должен знать разработчик на языке python, работая в сфере Data Science // E-Scio. 2020. №8 (47). URL: https://cyberleninka.ru/article/n/chto-dolzhen-znat-razrabotchik-na-yazyke-python-rabotaya-v-sfere-data-science (дата обращения: 04.04.2023).

8. Чибирова, М.Э. Анализ данных и регрессионное моделирование с применением языков программирования Python и R / Чибирова Марина Эльбрусовна // Научные записки молодых исследователей. 2019. №2. URL: https://cyberleninka.ru/article/n/analiz-dannyh-i-regressionnoe-modelirovanie-s-primeneniem-yazykov-programmirovaniya-python-i-r (дата обращения: 22.09.2022).

9. Li, S. Research on orthopedic auxiliary classification and prediction model based on XGBoost algorithm / Shenglong Li, Xiaojing Zhang // Neural Computing and Applications: Deep Learning & Neural Computing for Intelligent Sensing and Control. – Springer Nature Switzerland AG, 2020. – №32. – pp. 1971–1979.

10. Bentéjac, C. A comparative analysis of gradient boosting algorithms / Candice Bentéjac, Anna Csörgő, Gonzalo Martínez-Muñoz // Artificial Intelligence Review. – Springer Nature Switzerland AG, 2021. – №54. – pp. 1937–1967.

11. Bernard, S. Forest-RK: A New Random Forest Induction Method / Simon Bernard, Laurent Heutte, Sébastien Adam // International Conference on Intelligent Computing ICIC 2008: Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence. – Springer-Verlag Berlin Heidelberg, 2008. – pp. 430–437.

12. Cascaded Random Forest for Fast Object Detection / Florian Baumann, Arne Ehlers, Karsten Vogt, Bodo Rosenhahn // Scandinavian Conference on Image

Analysis SCIA 2013: Image Analysis. – Springer-Verlag Berlin Heidelberg, 2013. – pp. 131-142.

13. Demir, S. An investigation of feature selection methods for soil liquefaction prediction based on tree-based ensemble algorithms using AdaBoost, gradient boosting, and XGBoost / Selçuk Demir, Emrehan Kutlug Sahin // Neural Computing and Applications. – Springer Nature Switzerland AG, 2023. – №35. – pp. 3173–3190

14. Devan, P. An efficient XGBoost–DNN-based classification model for network intrusion detection system / Preethi Devan, Neelu Khare // Neural Computing and Applications. – Springer Nature Switzerland AG, 2020. – №32. – pp. 12499–12514.

15. Lasota, T. Investigation of Random Subspace and Random Forest Methods Applied to Property Valuation Data / Tadeusz Lasota, Tomasz Łuczak, Bogdan Trawiński // International Conference on Computational Collective Intelligence ICCCI 2011: Computational Collective Intelligence. Technologies and Applications. – Springer-Verlag Berlin Heidelberg, 2011. – pp. 142–151.

16. Lasota, T. Investigation of Random Subspace and Random Forest Regression Models Using Data with Injected Noise / Tadeusz Lasota, Zbigniew Telec, Bogdan Trawiński, Grzegorz Trawiński // International Conference on Knowledge-Based and Intelligent Information and Engineering Systems KES 2012: Knowledge Engineering, Machine Learning and Lattice Computing with Applications. – Springer-Verlag Berlin Heidelberg, 2013. – pp. 1–10.

17. Liu, X. Random Decision DAG: An Entropy Based Compression Approach for Random Forest / Xin Liu, Xiao Liu, Yongxuan Lai, Fan Yang, Yifeng Zeng // International Conference on Database Systems for Advanced Applications DASFAA 2019: Database Systems for Advanced Applications. – Springer-Verlag Berlin Heidelberg, 2019. – pp. 319–323.

18. Möhle, S. Modeling a System for Decision Support in Snow Avalanche Warning Using Balanced Random Forest and Weighted Random Forest / Sibylle

Möhle, Michael Bründl, Christoph Beierle // International Conference on Artificial Intelligence: Methodology, Systems, and Applications AIMSA 2014. – Springer-Verlag Berlin Heidelberg, 2014. – pp. 80–91.

19. Mohril, R.S. XGBoost based residual life prediction in the presence of human error in maintenance / Ram S. Mohril, Bhupendra S. Solanki, Makarand S. Kulkarni, Bhupesh K. Lad // Neural Computing and Applications: Applications of Machine Learning in Maintenance Engineering and Management (IFAC AMEST 2020). – Springer Nature Switzerland AG, 2023. – №35. – pp. 3025–3039.

20. Nguyen H. A novel whale optimization algorithm optimized XGBoost regression for estimating bearing capacity of concrete piles / Hieu Nguyen, Minh-Tu Cao, Xuan-Linh Tran, Thu-Hien Tran, Nhat-Duc Hoang // Neural Computing and Applications. – Springer Nature Switzerland AG, 2023. – №35. – pp. 3825-3852.

21. Torlay, L. Machine learning–XGBoost analysis of language networks to classify patients with epilepsy / L. Torlay, M. Perrone-Bertolotti, E. Thomas, M. Baciu // Brain Informatics. – Springer Nature Switzerland AG, 2017. – №4. – pp. 159–169.