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)

09.03.03 Прикладная информатика
(код и наименование направления подготовки, специальности)

Корпоративные информационные системы
(направленность (профиль)/ специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка программного модуля для предсказания уровня популярности объявлений о сдаче квартир в аренду»

Студент С.С. Курбонзода
(И.О. Фамилия) (личная подпись)

Руководитель Н.Н. Казаченок
(ученая степень, звание, И.О. Фамилия)

Консультант А.В. Москалюк
(ученая степень, звание, И.О. Фамилия)

Тольятти 2021

Аннотация

Тема бакалаврской работы: «Разработка программного модуля для предсказания уровня популярности объявлений о сдаче квартир в аренду».

В данной бакалаврской работе исследуются алгоритмы построения классификационных моделей, основанные на концепции деревьев решений (Decision Tree, Random Forest, Gradient Boosting и XGBoost).

На языке Python разработано программное обеспечение, реализующее предсказания класса популярности недвижимости (high, medium, low) на основе разнородного содержания объявления о квартире: текстовом описании, фотографиях, количестве спален, цене и т.д. Данные поступают с сайта по аренде квартир (renthop.com).

Бакалаврской работа состоит из введения, трёх глав, заключения и списка литературы. Во введении описывается актуальность проводимого исследования, дается краткая характеристика проделанной работы. В первой главе проводится обзор способов практического применения алгоритмов машинного обучения. Во второй главе описывается задача классификации, а также проводится сравнение алгоритмов машинного обучения, основанных на концепции деревьев решений. В третьей главе описывается программная реализация построения классификационных моделей (Decision Tree, Random Forest, Gradient Boosting и XGBoost) на данных объявлений аренды жилья и сравнивается их точность классификации на тестовых данных.

В заключении представлены выводы по проделанной работе.

В работе присутствуют 1 таблица, 47 рисунков. Список литературы состоит из 21 литературного источника. Общий объем выпускной квалификационной работы составляет 56 страниц.

Abstract

The topic of the bachelor's thesis is "Developing a software module to predict the level of popularity of apartments for rent".

In this bachelor's thesis we study algorithms for building classification models based on the concept of decision trees (Decision Tree, Random Forest, Gradient Boosting and XGBoost).

In Python, we developed software that implements predictions of the popularity class of a property (high, medium, low) based on the heterogeneous content of an apartment ad: text description, photos, number of bedrooms, price, etc. The data comes from an apartment rental website (renthop.com).

The bachelor's thesis consists of an introduction, three chapters, a conclusion and a list of references.

The introduction describes the relevance of the conducted research, gives a brief description of the work done.

The first chapter reviews the ways in which machine learning algorithms are used in practice.

The second chapter describes the classification problem and compares machine learning algorithms based on the concept of decision trees.

Chapter three describes the software implementation of building classification models (Decision Tree, Random Forest, Gradient Boosting and XGBoost) on housing advertisement data and compares their classification accuracy on test data.

The conclusion of the work is presented in the conclusion.

There are 1 table, 47 figures in the work. List of references consists of 21 literary sources. The total volume of the graduate qualification work is 56 pages.

Оглавление

Введение

В данной бакалаврской работе рассматривается деятельность компании по аренде жилья (сайт renthop.com). На сайте компании размещена информация о квартирах, которые можно арендовать. В объявлениях приведена различная информация о квартирах (дата размещения объявлений, описание квартир, фотографии). В течение длительного времени данная компания накапливала статистическую информацию об объявлениях и их популярности. Данная информация выложена в открытый доступ на сайте kaggle.com.

Пользователи сайта об аренде недвижимости, формируя запрос через веб-форму, получают список объявлений, отсортированный по дате создания объявлений. Для того чтобы повысить вероятность аренды жилья через сайт компании (а, следовательно, и прибыль компании) предлагается сортировать объявления по прогнозируемой популярности.

Актуальность темы исследования состоит в разработке классификатора популярности объявлений на исторических данных, предоставленных компанией по аренде жилья. Подобных систем разработано мало. Разработанный программный модуль позволит значительно сократить трудовые и временные затраты на прогнозирование популярности объявлений, что позволит упростить и ускорить работу компании.

Таким образом, цель работы – разработка программного модуля обучения классификатора для прогнозирования популярности объявлений.

Объект – процессы поиска и вывода объявлений по аренде квартир.

Предмет – автоматизация процесса вывода объявлений удовлетворяющих запросу пользователю с учетом прогнозируемой популярности.

Поставленная цель в работе достигается за счет решения следующих задач:

- анализ бизнес-процессов компании по аренде жилья для обоснования необходимости прогнозирования популярности объявлений;
- описание прогнозирование популярности объявлений как задачи классификация;
- разработка программного обеспечения для обучения классификатора популярности объявлений с использованием различных алгоритмов машинного обучения, основанных на концепции деревьев решений;
- определение алгоритма машинного обучения, обеспечивающего максимальную точность работы классификатора на имеющихся данных.

Бакалаврской работа состоит из введения, трёх глав, заключения и списка литературы.

Во введении содержатся краткие сведения о работе компании по организации аренды недвижимости. Отражены актуальность темы, цели и задачи бакалаврской работы.

В первой главе проводится анализ предметной области. Целью аналитической части является рассмотрение существующего состояния рынка недвижимости, характеристики их объекта и аппарата управления, выявления проблем и недостатков в работе систем и обоснование предложений по устранению выявленных недостатков, внедрению новых подходов, новых технологий.

Во второй главе описывается задача классификации, а также проводится сравнение алгоритмов машинного обучения, основанных на концепции деревьев решений.

В третьей главе описывается программная реализация построения классификационных моделей (Decision Tree, Random Forest, Gradient Boosting и XGBoost) на данных объявлений аренды жилья и сравнивается их точность классификации на тестовых данных. На языке Python разработано

программное обеспечение, реализующее предсказания класса популярности недвижимости (high, medium, low) на основе разнородного содержания объявления о квартире: текстовом описании, фотографиях, количестве спален, цене и т.д. Данные поступают с сайта по аренде квартир (renthop.com).

В заключении сделаны основные выводы по результатам бакалаврской работы, определено, какие задачи были решены, определены пути их внедрения и направления дальнейшего совершенствования работы фирмы.

Общий объем выпускной квалификационной работы составляет 56 страниц и включает 47 рисунков, 1 таблицу, 21 источник.

Глава 1 Функциональное моделирование предметной области

1.1 Характеристика деятельности агентства недвижимости

В настоящее время агентства по аренде жилья играют значительную роль на рынке недвижимости. Агентство по аренде жилья осуществляет посредничество между двумя сторонами покупателем и арендодателем, отслеживая юридическую сторону сделки, являясь гарантом безопасности. Агентства по аренде жилья подбирают квартиру в соответствии с требованиями покупателя, отслеживают ее историю, снижают до минимума риски обеих сторон при осуществлении сделки и выступают посредниками между продавцом и покупателем.

Качество работы агентства определяет регулярная отчетность о проделанной работе, определенность во времени, конкретика, актуальность его предложений.

За последние двадцать лет значительно возрос объём и оборот информации во всех сферах жизнедеятельности человека: экономической, финансовой, политической, духовной. И процесс накопления, обработки и использования знаний постоянно ускоряется. Учёные утверждают, что каждые десять лет количество информации увеличивается вдвое. В связи с этим возникает необходимость использования автоматических средств, позволяющих эффективно хранить, обрабатывать и распределять накопленные данные.

Исходя из современных требований, предъявляемых к качеству работы финансового звена крупного предприятия, эффективная работа предприятия зависит от уровня оснащения компании информационными средствами на базе компьютерных систем.

1.2 Выбор технологии концептуального моделирования исследуемой области

IDEF0 – функционального моделирования (англ. function modeling) и графическая нотация, предназначенная для формализации и описания бизнес-процессов. Отличительной особенностью IDEF0 является её акцент на соподчинённость объектов. В IDEF0 рассматриваются логические отношения между работами, а не их временная последовательность (поток работ).

Стандарт IDEF0 представляет организацию как набор модулей, здесь существует правило – наиболее важная функция находится в верхнем левом углу, кроме того, существуют правила сторон:

- стрелка входа всегда приходит в левую кромку активности;
- стрелка управления — в верхнюю кромку;
- стрелка механизма — нижняя кромка;
- стрелка выхода — правая кромка.

Описание выглядит как «чёрный ящик» с входами, выходами, управлением и механизмом, который постепенно детализируется до необходимого уровня. Также для того чтобы быть правильно понятым, существуют словари описания активностей и стрелок. В этих словарях можно дать описания того, какой смысл вы вкладываете в данную активность либо стрелку.

Описание методологии IDEF0 содержится в рекомендациях Р 50.1.028-2001 «Информационные технологии поддержки жизненного цикла продукции. Методология функционального моделирования».

Также отображаются все сигналы управления, которые на DFD (диаграмме потоков данных) не отображались. Данная модель используется при организации бизнес-процессов и проектов, основанных на моделировании всех процессов: как административных, так и организационных.

1.3 Моделирование бизнес-процессов предметной области для постановки задачи автоматизированного варианта решения

Компания Renthop занимается сбором, обработкой и хранением базе данных на сервере информации о жилье, которое собственники сдают в аренду. Пользователи могут искать информацию о жилье посредством сайта renthop.com. Пользователь сайта может искать квартиры, удовлетворяющие его запросу. Список найденных квартир будет отсортирован по дате размещения объявлений и выведен на экран в виде веб-страницы (рисунок 1).

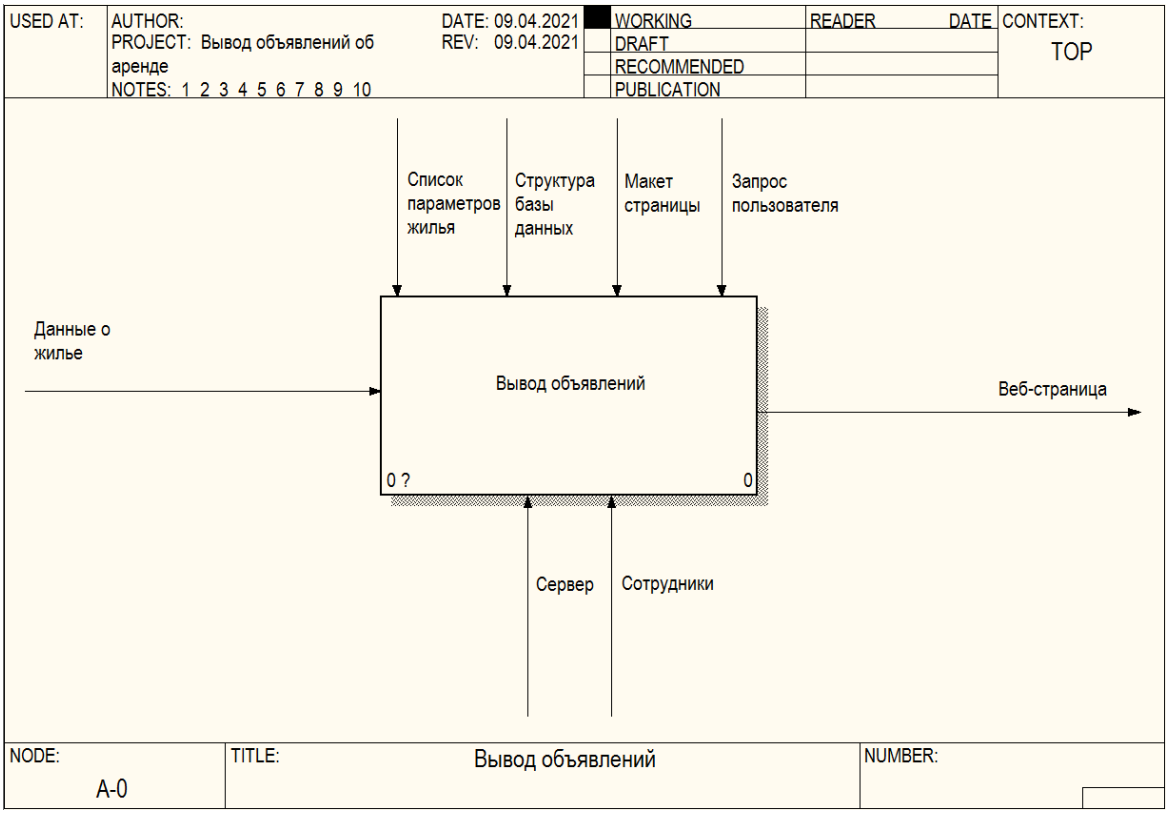

Рисунок 1– Контекстная диаграмма «Вывод объявлений» для модели «Как есть»

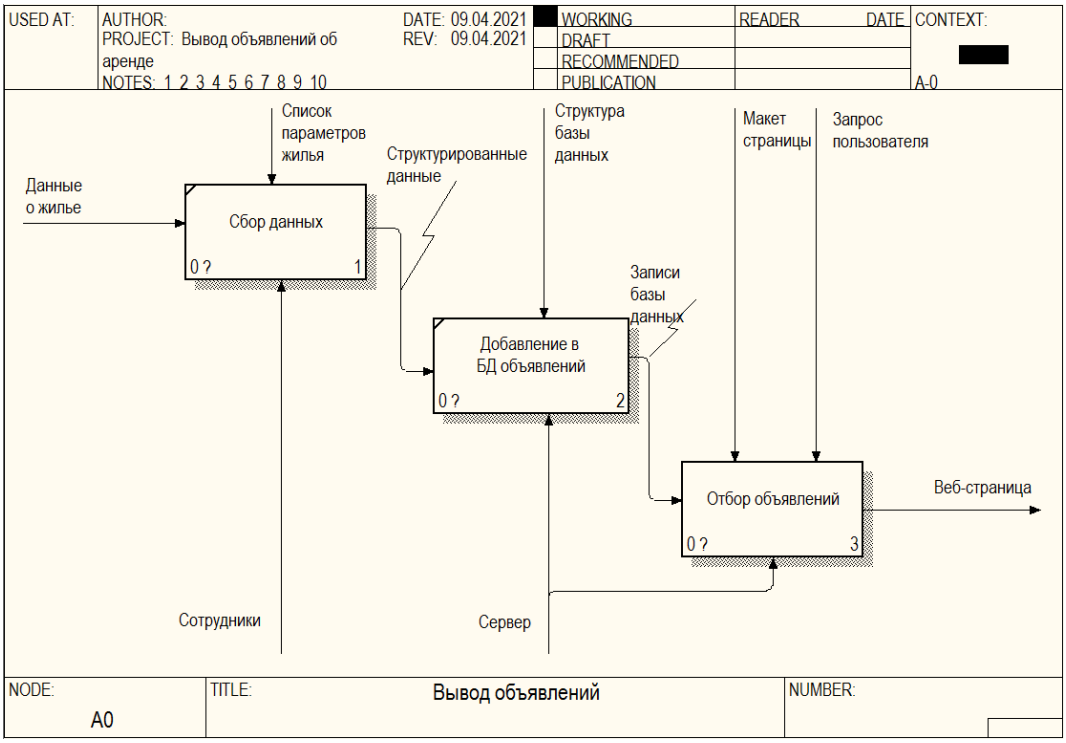

Рисунок 2 – Диаграмма «Вывод объявлений» (A0) для модели «Как есть»

Дальнейший анализ деятельности компании позволил произвести декомпозицию процесса «Вывод объявлений». Диаграмма «Вывод объявлений» (A0) для модели «Как есть» показана на рисунке 2.

Потенциалом для усовершенствования является процесс «Вывод объявлений». Пользователь сайта об аренде недвижимости, формируя запрос через веб-форму, получают список объявлений, отсортированный по дате создания объявлений (рисунок 3). Это приводит к тому, что интересные и, в будущем, востребованные (популярные) объявления оказываются на последних страницах с результатами поиска. Поэтому процесс «Отбор объявлений» предлагается изменить следующим образом. Предлагается на основе исторических данных компании renthop обучить классификатор (работающий на сервере компании) способный прогнозировать популярность объявлений. При этом сортировку найденных в соответствии с запросом пользователя объявлений производить с учетом прогнозируемой популярности объявлений (рисунок 4).

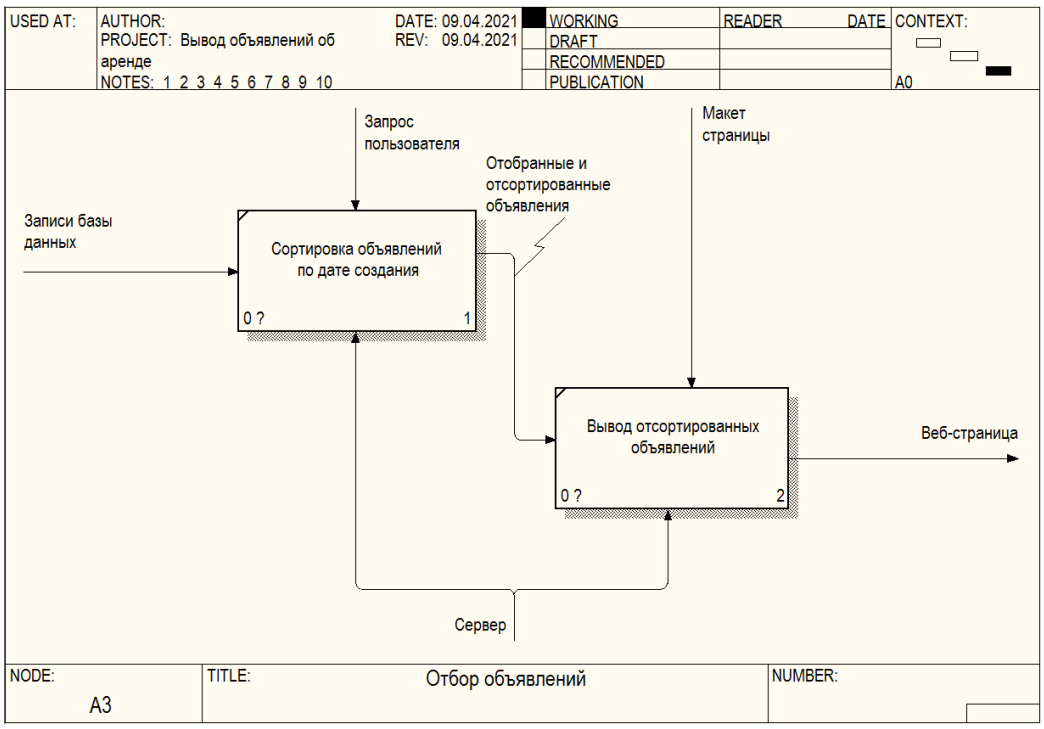

Рисунок 3 – Диаграмма «Отбор объявлений» (A0) для модели «Как есть»

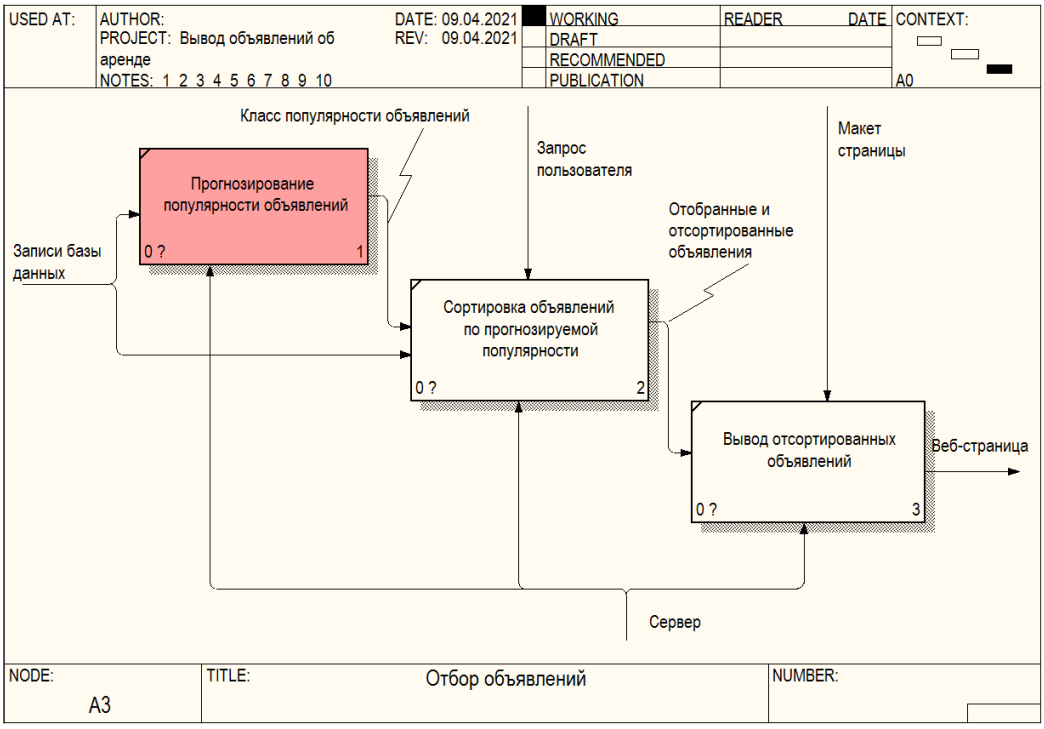

Рисунок 4 – Диаграмма «Отбор объявлений» (A0) для модели «Как должно быть»

Таким образом, усовершенствование исследуемого бизнес-процесса достигается путём внедрения программного обеспечения для прогнозирования популярности объявлений о жилье.

1.4 Постановка задачи на разработку программной реализации и ее внедрения в деятельность компании

Процесс автоматизации процесса вывода объявлений удовлетворяющих запросу пользователю с учетом прогнозируемой популярности приведет к уменьшению временных затрат на оперативную обработку информации при поиске нужного объекта недвижимости.

Тип проектируемой программы – диалоговая программа решения задачи, которая позволит максимально быстро обрабатывать нужную информацию для вывода объявлений.

С помощью программы улучшим следующие значения качества обработки информации:

- сокращение времени обработки информации;
- сокращение времени на поиск нужного объекта недвижимости;
- повышение степени достоверности обработки информации;
- исключение времени на поиск «вручную».

Выполнение поставленной задачи позволит сократить значительную часть времени для работников компании.

1.5 Общая характеристика организации решения задачи на ЭВМ

Программный модуль, реализующий предсказания класса популярности недвижимости на основе разнородного содержания объявления о квартире позволит значительно упростить процесс подбора объявлений в агентстве недвижимости, сократить трудовые и временные

затраты на ведение документации, что позволит упростить и ускорить работу агентства недвижимости.

Основными источниками оперативной информации являются данные, которые поступают с сайта по аренде квартир, предоставляемые в агентство недвижимости для работы с клиентом, а так же сведения о необходимом запросе на поиск, аренду или обмен объекта недвижимости.

Программный модуль будет разработан на языке Python и будет представлять собой программу, работающую на основе базы данных сайта по аренде квартир (renthop.com), где представлены разнородного содержания объявления о квартире: текстовом описании, фотографиях, количестве спален, цене и т.д.

Выводы по первой главе

В качестве выводов можно отметить следующее.

По результатам материалов, представленных в данной главе, была поставлена задача автоматизации процесса вывода объявлений удовлетворяющих запросу пользователю. При этом усовершенствование исследуемого бизнес-процесса достигается путём разработки и внедрения программного обеспечения для прогнозирования популярности объявлений. Информация о прогнозируемой популярности используется для сортировки объявлений удовлетворяющих запросам пользователя.

Глава 2 Проектирование программного модуля

2.1 Прогнозирование популярности объявлений как задача классификации

Задача классификации (classification task) отличается тем, что в ней существует множество допустимых ответов, называемых метками классов, и оно конечно. В данном случае под классом понимается множество объектов с данным значением метки.

Решение задачи классификации позволяет выводить закономерности и некоторые зависимости из наблюдений и прецедентов в имеющихся у нас данных. Такая постановка задачи есть обобщение задач аппроксимации функций.

Целью задачи классификации является определение принадлежности нового наблюдения к одному или нескольким из множества классов. Определение основано на обучающем наборе данных, содержащем наблюдения, чьи классы уже определены.

Перейдем к формальному определению задачи классификации.

Имеется обучающая выборка X, как множество отдельных x_i объектов, обозначаемая формулой:

$$X = \{x_i\}_{i=1}^{n}. \quad (1)$$

Переменная y (называемая также target value, переменная отклика) фигурирует в качестве исхода объекта x и принимает конечное число значений. Необходимо построить классификатор, который по вектору признаков x возвратит метку класса $\hat{y}$ или вектор оценок, характеризующий вероятность принадлежности к каждому из классов.

Задача классификации относится к блоку задач обучения с учителем и включает в себя 2 основных этапа:

- этап тренировки модели: извлечение признаков для модели из тренировочного набора данных, обучение модели;
- этап предсказания: извлечение признаков для модели из тестового набора данных, передача данных в обученную модель, получение отклика модели.

В случае если количество классов больше 2, говорят о решении задачи мульти-классификации.

Остановимся подробней на наиболее часто используемых алгоритмах машинного обучения, предназначенных для решения задач классификации.

Общая схема эволюции алгоритмов представлена на рисунке 5.

Рисунок 5 – Схема эволюции описанных алгоритмов

Сравнение алгоритмов и их отличительные особенности показаны в таблице 1.

Таблица 1 – Отличие алгоритмов, основанных на построении деревьев принятия решений

Название алгоритма	**Отличия по сравнению с предыдущим алгоритмом**
Алгоритмы построения одиночных деревьев принятия решений (Decision Tree's algorithm)	Графическое представление возможных решений (в виде дерева), основанных на определенных условиях.
Бэггинг (Bagging)	Bootstrap-агрегирование или Bagging - это ансамблевый мета-алгоритм, основанный на учете предсказания от нескольких деревьев решений и использующий механизм голосования.
Случайный лес (Random Forest)	Алгоритм, основанный на пакетировании данных. Исходные данные случайным образом делятся на подмножества и, для каждого подмножества, строится свое дерево принятия решений.
Бустинг (Boosting)	Деревья решения строятся последовательно. Каждое последующее дерево направлено на минимизацию ошибки предыдущих моделей
Градиентный бустинг (Gradient Boosting)	Для минимизации ошибок в последовательных моделях используется алгоритм градиентного спуска.
XGBoost	Оптимизированная версия алгоритма Gradient Boosting за счет параллельной обработки, обрезки деревьев, обработки пропущенных значений и регуляризации.

При решении задачи классификации объявлений об аренде будем использовать алгоритмы Decision Tree, Random Forest, Gradient Boosting и XGBoost и производить сравнение их точности работы.

2.2 Описание набора данных

Файловая структура данных об объявлениях аренды, предоставляемая сайтом renthop.com выглядит следующим образом:

- train.json – тренировочный набор данных в формате json (здесь содержатся данные, которые можно представить в текстовом виде, фотографии содержаться в отдельном архиве);

- test.json – тестовый набор данных в формате json (здесь находятся данные по которым производится тестирование точности классификатора);
- sample_submission.csv – файл содержащий несколько записей исходных данных в формате csv;
- images_sample.zip – примеры фотографий квартир, связанные с данными в текстовом формате (содержащихся в файле sample_submission.csv) через поле listing_id;
- renthop.7z – все фотографий квартир, связанные с данными в текстовом формате (содержащихся в файлах train.json и test.json) через поле listing_id.

Таблица исходных данных содержит в себе следующие поля (столбцы):

- bathrooms – количество ванных-комнат;
- bedrooms – количество спален;
- building_id – ID здания в котором находиться жилье;
- created – дата создания объявления;
- description – описание объявления;
- display_address – отображаемый в объявлении адрес жилья;
- features – список особенностей жилья;
- latitude – широта расположения жилья (используется для отображения на интерактивной карте);
- listing_id – ID для привязки к объявлению фотографий;
- longitude – долгота расположения жилья (используется для отображения на интерактивной карте);
- manager_id – ID менеджера;
- photos – список ссылок на фотографии с сайта renthop.com (по ссылкам находятся те же фотографии, которые содержатся в архиве renthop.7z);
- price – цена аренды в долларах;
- street_address – название улицы на которой находится жилье;

- interest_level – метка класса объявления, обозначающая уровень интереса к объявлению (возможно одно из трех значений «high», «medium», «low»).

Следовательно, описанная файловая структура набора данных, предоставляемых сайтом по аренде квартир, позволит решить задачу классификации объявлений об аренде.

Выводы по второй главе

По результатам описанных исследований можно сделать следующие выводы, в частности, прогнозирование популярности объявлений о сдаче квартир в аренду можно представить в виде задачи классификации. Тогда для прогнозирования требуется обучить классификатор, способный на основе входных параметров объявлений, определять класс его популярности.

Рассмотрены алгоритмы машинного обучения, способные решать задачи классификации и основанные на использовании деревьев принятия решений. На основе их различий и особенностей составлена сводная таблица.

Описан набор данных предоставляемых сайтом по аренде квартир (renthop.com), объяснены значения входных параметров и выходных меток класса. А также описана файловая структура набора данных.

Глава 3 Разработка программного обеспечения для анализа данных

3.1 Описание средств разработки и программного кода модуля

На языке Python разработано программное обеспечение, реализующее предсказания класса популярности недвижимости (high, medium, low) на основе разнородного содержания объявления о квартире: текстовом описании, фотографиях, количестве спален, цене и т.д. Данные поступают с сайта по аренде квартир (renthop.com). Анализируемые данные агентства по аренде квартир доступно также по ссылке - https://www.kaggle.com.

Для работы с данными будем использовать следующие стандартные библиотеки:

- json – библиотека содержащая методы для работы с данными в формате json;
- numpy – библиотека, содержащая набор метаматематических функций и добавляющая возможность работы с массивами;
- pandas – библиотека содержащая методы для работы с табличными данными;
- matplotlib – библиотека содержащая в себе методы для визуализации данных.

Подключение библиотек показано на рисунке 6.

Импорт необходимых библиотек

```
In [1]: import json

In [2]: import numpy as np
        import pandas as pd

In [3]: import matplotlib.pyplot as plt
        %matplotlib inline
```

Рисунок 6 – Подключение библиотек к проекту

Для того чтобы загрузить имеющееся множество данных воспользуемся инструкцией open, укажем название и расположение файла с тренировочными данными в формате json. Впоследствии данные загружаются из потока f в переменную data типа dict (словарь) (рисунок 7).

Загрузка тренировочных данных

```
In [4]: with open('train.json') as f:
            data = json.load(f)
```

Рисунок 7– Загрузка данных из файла train.json

Для того чтобы посмотреть, какие признаки содержаться в данных воспользуемся методом keys() и, для удобства вывода, конвертируем результат в тип данных list (рисунок 8).

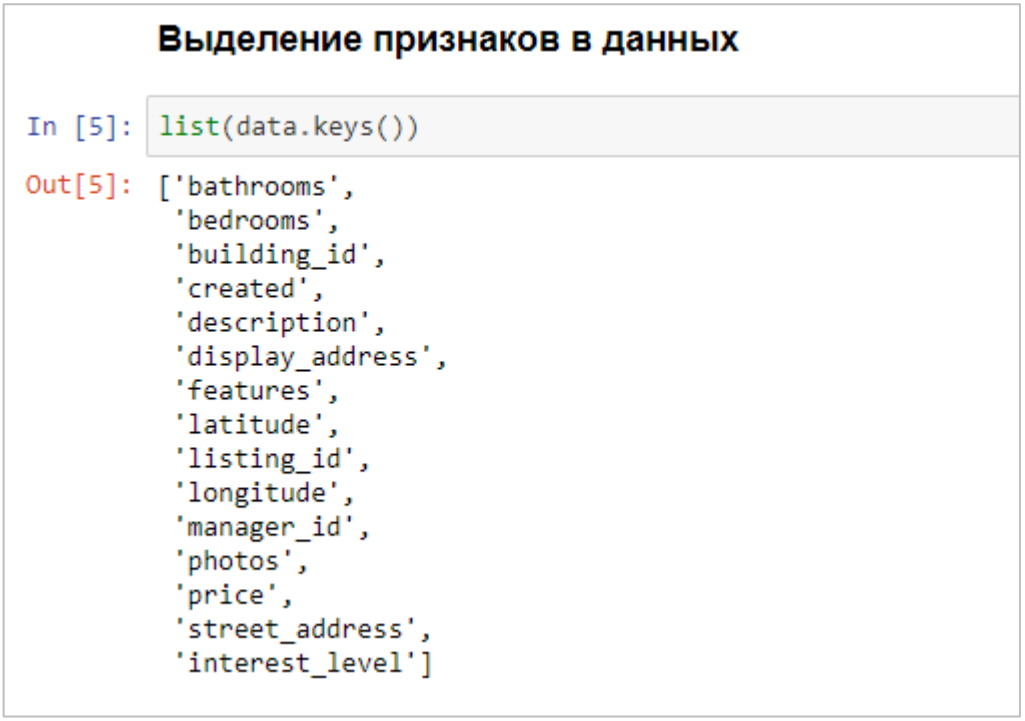

Рисунок 8 – Извлечение признаков в данных

Описание значений каждого признака приведено в разделе 2.3.

Так как с переменными типа dict при анализе данных работать неудобно осуществим переход к использованию другого типа переменных – dataframe. Это особый тип переменных для работы с данными, которые можно представить в табличном виде. К тому же, dataframe имеют базовые встроенные методы для анализа хранимых в них данных. В библиотеке pandas, обеспечивающей работу с dataframe, есть встроенный метод from_dict, обеспечивающий конвертирование данных в dataframe из переменных типа dict.

Воспользуемся этим методом, чтобы сформировать переменную df (тип dataframe) содержащую наш набор данных (рисунок 9). Также определим индексы dataframe как значения типа int, чтобы не возникало проблем при работе со встроенными методами pandas.

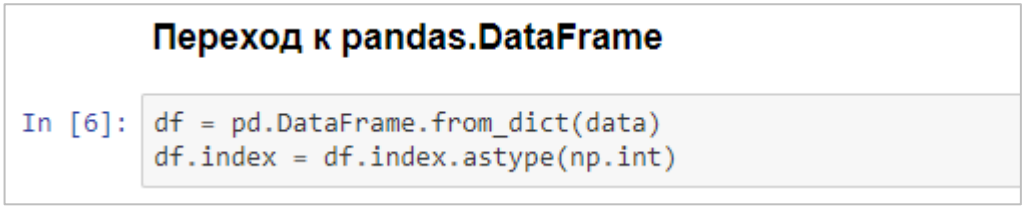

Рисунок 9 – Конвертирование данных из типа dict в тип dataframe

Оценим размерность исходных данных. Для этого воспользуемся командой df.shape так как это показано на рисунке 10. Как видим, в нашем наборе данных содержится информация о 49352 объявлениях об аренде и каждое объявление описывается 15 параметрами. Все эти данные будут использованы для обучения классификатора объявлений.

Рисунок 10 – Определение размерности данных

Теперь осуществим сортировку данных по индексу, а также выведем первые 5 строк данных из df, воспользовавшись методом head(). Результат выполнения данных действий показан на рисунке 11.

```
In [9]: df.head()
```

Out[9]:

	bathrooms	bedrooms	building_id	created	description	display_a
4	1.0	1	8579a0b0d54db803821a35a4a615e97a	2016-06-16 05:55:27	Spacious 1 Bedroom 1 Bathroom in Williamsburg!...	145 Bor
6	1.0	2	b8e75fc949a6cd8225b455648a951712	2016-06-01 05:44:33	BRAND NEW GUT RENOVATED TRUE 2 BEDROOMFind you...	Ea
9	1.0	2	cd759a988b8f23924b5a2058d5ab2b49	2016-06-14 15:19:59	**FLEX 2 BEDROOM WITH FULL PRESSURIZED WALL**L...	East 56th
10	1.5	3	53a5b119ba8f7b61d4e010512e0dfc85	2016-06-24 07:54:24	A Brand New 3 Bedroom 1.5 bath ApartmentEnjoy ...	Metro
15	1.0	0	bfb9405149bfff42a92980b594c28234	2016-06-28 03:50:23	Over-sized Studio w abundant closets. Availabl...	East 34th

Рисунок 11 – Вывод нескольких объявлений из dataframe (переменной df)

Выведем типы всех параметров, описывающих каждое объявление. Это действие необходимо для того, чтобы понять, каким образом хранятся значения в dataframe. Для этого воспользуемся командой df.dtype. Результат показан на рисунке 12. Как видно, при конвертации данных встроенный в pandas метод корректно определил типы данных для каждого параметра объявления.

```
In [10]: df.dtypes

Out[10]: bathrooms          float64
         bedrooms             int64
         building_id         object
         created             object
         description         object
         display_address     object
         features            object
         interest_level      object
         latitude           float64
         listing_id           int64
         longitude          float64
         manager_id          object
         photos              object
         price                int64
         street_address      object
         dtype: object
```

Рисунок 12 – Вывод типа данных для каждого параметра таблицы с объявлениями об аренде из переменной df

Для того чтобы посмотреть такие, как характеристики, как количество, среднее значение, стандартное отклонение, минимально и максимальное значения, значения трех квантилей для количественных признаков (типа int и float64) воспользуемся встроенным методом describe(). Результат показан на рисунке 13.

```
In [11]: df.describe()
```

Out[11]:

	bathrooms	bedrooms	latitude	listing_id	longitude	price
count	49352.00000	49352.000000	49352.000000	4.935200e+04	49352.000000	4.935200e+04
mean	1.21218	1.541640	40.741545	7.024055e+06	-73.955716	3.830174e+03
std	0.50142	1.115018	0.638535	1.262746e+05	1.177912	2.206687e+04
min	0.00000	0.000000	0.000000	6.811957e+06	-118.271000	4.300000e+01
25%	1.00000	1.000000	40.728300	6.915888e+06	-73.991700	2.500000e+03
50%	1.00000	1.000000	40.751800	7.021070e+06	-73.977900	3.150000e+03
75%	1.00000	2.000000	40.774300	7.128733e+06	-73.954800	4.100000e+03
max	10.00000	8.000000	44.883500	7.753784e+06	0.000000	4.490000e+06

Рисунок 13 – Характеристики параметров типа int и float64

Обязательным шагом предварительного анализа данных является поиск пропущенных значения (на языке python такие значения обозначаются как NaN). Не все алгоритмы машинного обучения позволяют работать с пропущенными значениями, поэтому об их наличии необходимо знать заранее. Анализ пропущенных значений осуществляется с помощью метода info(). Результат его использования на наших данных показан на рисунке 14.

Как видно из результатов, показанных на рисунке 14, в наших данных отсутствуют пропущенные значения (т.е. в каждом столбце по 49352 значения, что соответствует размеру таблицы исходных данных). Это означает, что нам не придется отбрасывать данные с пропущенными значениями или определять стратегию заполнения пропусков, что облегчает задачу обучения классификатора популярности объявлений об аренде жилья.

Поиск NaN значений

```
In [12]: df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 49352 entries, 4 to 124009
Data columns (total 15 columns):
bathrooms          49352 non-null float64
bedrooms           49352 non-null int64
building_id        49352 non-null object
created            49352 non-null object
description        49352 non-null object
display_address    49352 non-null object
features           49352 non-null object
interest_level     49352 non-null object
latitude           49352 non-null float64
listing_id         49352 non-null int64
longitude          49352 non-null float64
manager_id         49352 non-null object
photos             49352 non-null object
price              49352 non-null int64
street_address     49352 non-null object
dtypes: float64(3), int64(3), object(9)
memory usage: 6.0+ MB
```

Рисунок 14 – Проверка данных на наличие пропущенных значений

Также в ходе предварительного анализа данных надо понять, существуют ли ситуации, когда одна и та же квартира описана в нескольких объявлениях. Если такие ситуации есть, то надо каким-то образом группировать объявления, относящиеся к одной квартире, для последующей предобработки данных.

Такой анализ можно осуществить, используя метод groupby с указанием названия колонки таблицы, по которой будет производиться группировка (в нашем случае это listing_id – ID объявления). Затем с помощью метода count() подсчитаем количество групп, в которых количество объявлений больше одного и результат выведем на экран (рисунок 15). В нашем случае нет ни одного случая создания нескольких объявлений для одной квартиры.

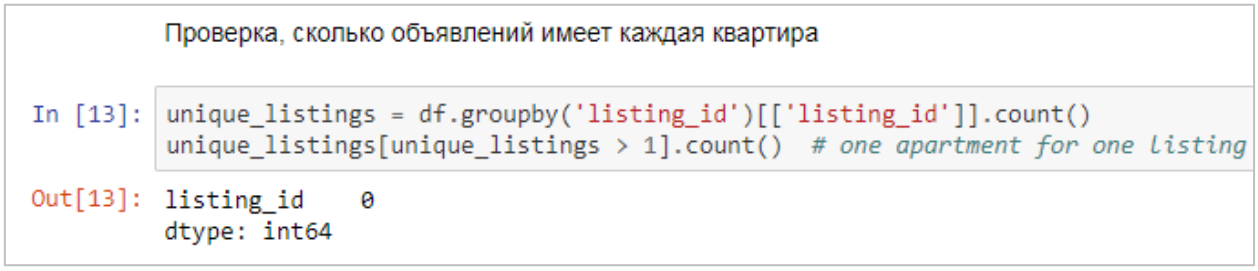

Рисунок 15 – Подсчет количества случаев, когда у одной квартиры несколько объявлений

Перейдем к этапу поиска выбросов (аномальных значений) в количественных признаках объявлений. Проанализируем данные, которые задаются произвольно при размещении объявлений о сдаче квартиру в аренду: количество ванных-комнат (bathroms), количество спален (bedrooms) и цену аренды (price). На основе имеющихся данных построим график с использованием встроенного в библиотеку Pandas метода plot(). При этом укажем тип отображаемого графика – «box». С помощью команды plt.show() выводим график на экран (рисунок 16).

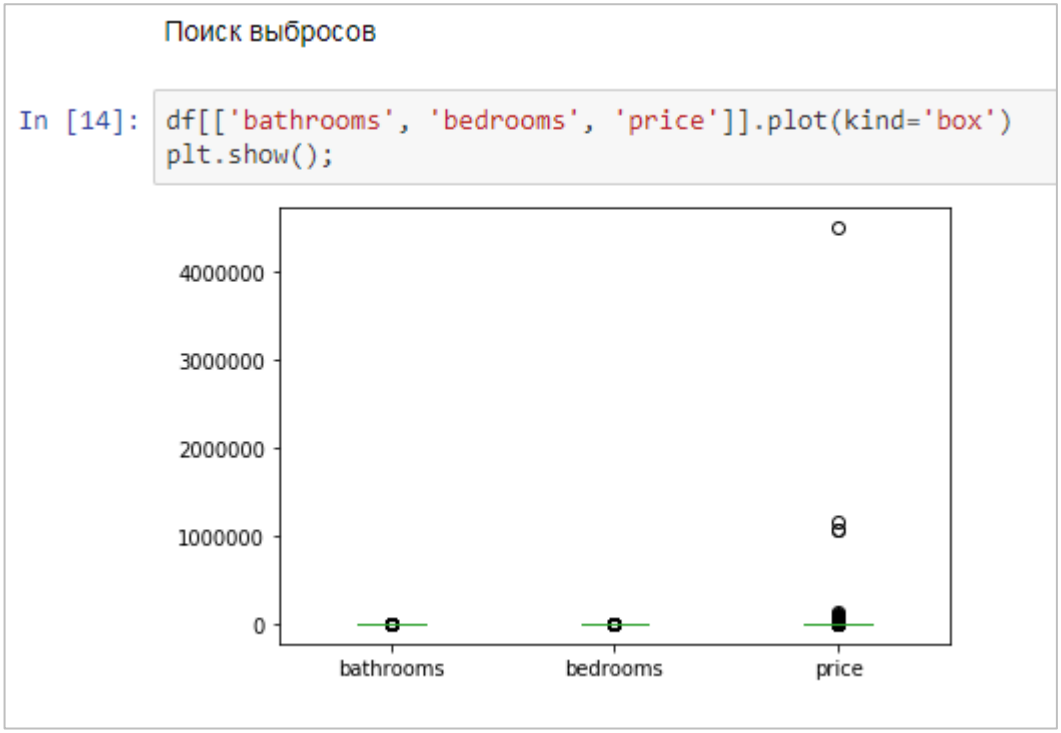

Рисунок 16 – Контроль выбросов в числовых значениях данных

На рисунке 16 видно, что существуют несколько объявлений, в которых цена аренды (price) на несколько порядков больше, чем в подавляющем большинстве объявлений. Выведем отдельно эти объявления с аномальными значениями цены аренды. Для этого воспользуемся условием df[`price`] > 1000000.0 так, как это показано на рисунке 17.

Как видно из рисунка 17 в нашей выборке данных присутствует всего 4 объявления с аномально высокой ценой аренды. Эти объявления обладают низкой популярностью, так как из-за неправильно заданной цены большинство пользователей их не просматривает.

Объявления с аномальными значениями не обладают значимостью для обучения классификатора объявления, поэтому их нужно отсеять перед дальнейшим анализом данных.

Для этого зададим переменную outliers_indexes типа list куда будут помещены индексы отсеиваемых объявлений. Программный код для заполнения переменной outliers_indexes показан на рисунке 18

Выбросы по цене

```
In [15]: df[df['price'] > 1000000.0]
```

Out[15]:

	created	description	display_address	features	interest_level	latitude	listing_id	longitude	
12168	2016-06-24 05:02:58		West 116th Street	[Doorman, Elevator, Cats Allowed, Dogs Allowed...	low	40.8011	7208794	-73.9480	d173
32611	2016-06-24 05:02:11		Hudson Street	[Doorman, Elevator, Cats Allowed, Dogs Allowed...	low	40.7299	7208764	-74.0071	d173
55437	2016-05-14 05:21:28		West 57th Street	[Doorman, Cats Allowed, Dogs Allowed]	low	40.7676	7013217	-73.9844	8f5
57803	2016-05-19 02:37:06	This 1 Bedroom apartment is located on a prime...	West 57th Street	[Doorman, Elevator, Pre-War, Dogs Allowed, Cat...	low	40.7676	7036279	-73.9844	18

Рисунок 17 – Объявления с аномальными числовыми значениями

```
In [16]: outliers_indexes = df[df['price'] > 1000000.0].index.tolist()
         outliers_indexes

Out[16]: [12168, 32611, 55437, 57803]
```

Рисунок 18 – Формирование списка индексов отсеиваемых объявлений

Теперь необходимо привести поле created (дата создания объявления) к правильному формату. Дело в том, что изначально при конвертации этого значения оно было определено как object (строковый тип). Однако на языке python для дат есть специальный тип данных – datetime, к которому нужно приводить временные метки, чтобы использовать дополнительный аналитический инструментарий.

Перевода строковых значений поля created во временной формат воспользуемся встроенным в библиотеку pandas методом to_datetime() в котором укажем название поля. Затем создадим новое поле, под названием timestamp, которое будет содержать только дату создания объявления (без указания времени, как в поле created). Программный код данного шага показан на рисунке 19.

Дата создания объявления (перевод в datetime формат)

```
In [17]: df['created'] = pd.to_datetime(df['created'])
         df['timestamp'] = df['created'].dt.date
```

Рисунок 19 – Конвертирования поля типа object в тип datatime

Теперь с помощью метода groupby() сгруппируем данные по значению timestamp, и визуализируем изменение количества появление новых объявлений. Реализуем это с помощью программного кода, указанного на рисунке 20.

```
In [18]: df.groupby('timestamp')[['timestamp']].count().plot()
         plt.show();
```

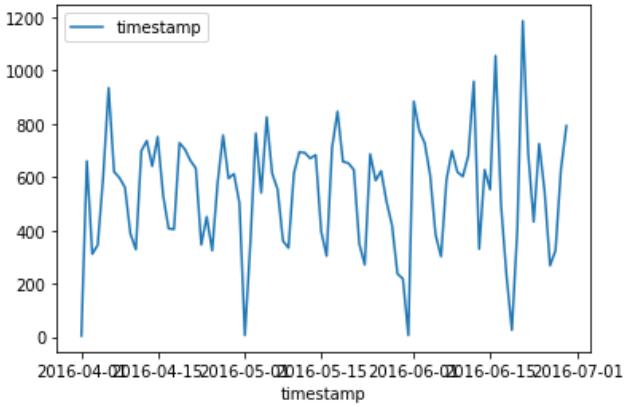

Рисунок 20 – Изменение количества (ось Y) появления новых объявлений в течение времени (ось X)

На популярность объявления может оказывать влияние день его создания, месяц создания и год создания. Логичным решением будет создание этих дополнительных полей (created_day, created_month, created_year) в нашей выборке данных, хранящейся в переменной df типа dataframe. Код создающий эти дополнительные поля показан на рисунке 21.

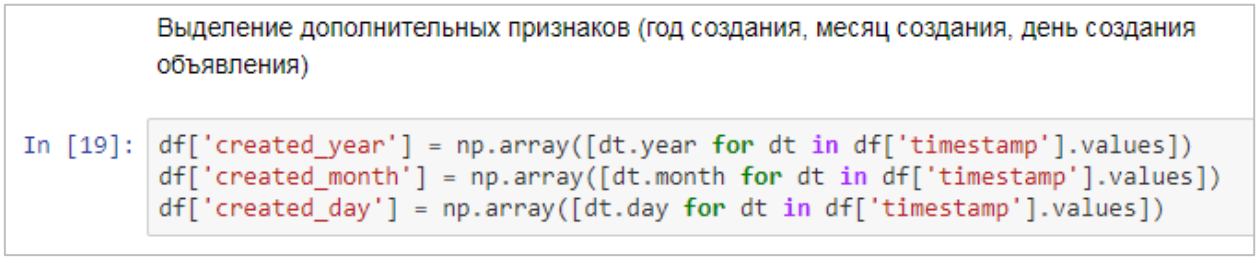

Рисунок 21 – Создание дополнительных временных признаков

Произведем предобработку качественных признаков, имеющихся в наших данных. Предобработка этих данных требуется, так как алгоритмы машинного обучения не умеют самостоятельно выбирать стратегию учета значений текстовых полей. К качественным признакам жилья относятся следующие поля: features – список особенностей жилья и description – текстовое описание жилья.

Посчитаем количество символов в каждом из этих полей, и для каждого объявления укажем получившиеся значения в столбцах features_qnt и description_qnt. Для этого воспользуемся кодом, показанным на рисунке 22.

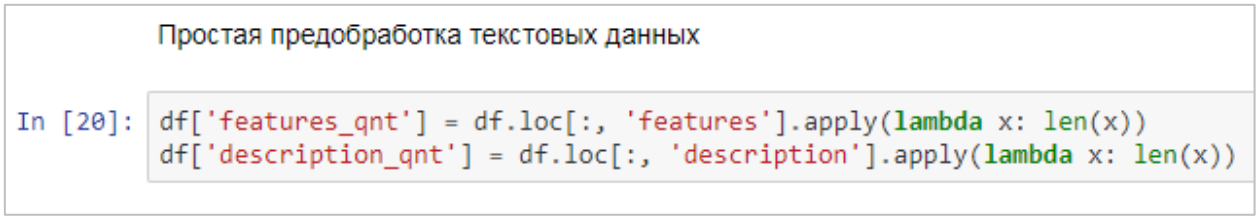

Рисунок 22 – Предобработка текстовых полей

Также можно провести дополнительную предобработку текстового поля features (список особенностей жилья) с помощью токенизации текста. Для этого воспользуемся экстрактором признаков из текста, встроенного в библиотеку sklearn, под название CountVectorizer. Модуль CountVectorizer позволяет сконвертировать набор текстов в матрицу токенов, находящихся в тексте. Он относится к концепции обработки естественных языков Bag-of-Words, анализируя количественное вхождение слов в документах. Применим CountVectorizer к нашему столбцу features. Однако пред этим требуется очистить хранящийся в поле текст от пробелов, заменив их при этом на подчеркивания. Это делается для того, чтобы, например, особенность «Cats allowed» преобразовалась в «Cats_allowed» и впоследствии рассматривалась как один токен «Cats_allowed», а не два отдельных: «Cats» и «allowed». Пример значений столбца features показан на рисунке 17.

Результат предобработки будем хранить в столбце под названием features_str. Затем создадим экземпляр CountVectorizer, поместив его в переменную tfidf, задав при этом язык исходного текста – английский (stop_words=’english’) и ограничив количество извлекаемых признаков (max_features=200).

Результат тонизации сохраним в переменную tr_sparse, указав в качестве источника предобработанные данные, хранящиеся в столбце features_str. При этом исключим все объявления с аномальными значениями – это те объявления, в которых цена аренды аномально высокая, они представлены на рисунке 17. Для исключения этих объявлений воспользуемся списком outliers_indexes. Программный код этого этапа показан на рисунке 23.

```
In [31]: df['features_str'] = df['features'].apply(lambda x: ' '.join(['_'.join(i.split(' ')) for i in x]))

In [32]: tfidf = CountVectorizer(stop_words='english', max_features=200)

In [33]: tr_sparse = tfidf.fit_transform(df['features_str'].drop(outliers_indexes, axis=0))

In [34]: tr_sparse.shape

Out[34]: (49348, 200)
```

Рисунок 23 – Предобработка текстовых полей

Теперь перейдем к анализу графической информации объявлений. Для выгрузки фотографий жилья, прикрепленных к объявлениям, воспользуемся библиотекой requests. Для отображения и анализа изображений воспользуемся библиотекой cv2. Код для подключения библиотек показан на рисунке 24.

```
In [35]: import requests
         import cv2
```

Рисунок 24 – Подключение библиотек для анализа изображений

В исходных данных к каждому объявлению прикреплен список веб-ссылок, по которым находятся фотографии жилья. Для того чтобы получить непосредственно изображения по этим ссылкам воспользуемся методом get() библиотеки requests. При этом в качестве параметра photo будем последовательно указывать ссылки, хранящиеся в столбце photos. Последовательная подстановка ссылок будет реализована посредством цикла for. Для того, чтобы обеспечить вывод изображения на экран необходимо воспользоваться методом imdecode(), который преобразует массив значений

nparr в графическое представление. В качестве второго входного параметра укажем формат отображения изображения, а именно cv2.imread_color (т.е. цветное изображение). С помощью стандартного метода print() также выведем разрешение всех загруженных изображений связанных с анализируемым объявлением. Программный код для загрузки изображений показан на рисунке 25.

С помощью инструкции plt.imshow(img_np) осуществляет вывод изображения пользователю. Программный код и результат вывода изображения показан на рисунке 26.

```
In [36]: for photo in df['photos'][4]:
             nparr = np.fromstring(requests.get(photo).content, np.uint8)
             img_np = cv2.imdecode(nparr, cv2.IMREAD_COLOR)
             print(img_np.shape)

C:\Users\user_admin\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: Deprec
ationWarning: The binary mode of fromstring is deprecated, as it behaves surpri
singly on unicode inputs. Use frombuffer instead

(640, 480, 3)
(640, 480, 3)
(640, 480, 3)
(640, 480, 3)
(640, 480, 3)
(640, 480, 3)
(640, 480, 3)
(640, 480, 3)
(640, 480, 3)
(640, 480, 3)
(640, 480, 3)
```

Рисунок 25 – Программный код для загрузки изображений по веб-ссылкам

```
In [37]: plt.imshow(img_np)
Out[37]: <matplotlib.image.AxesImage at 0x17de8a20>
```

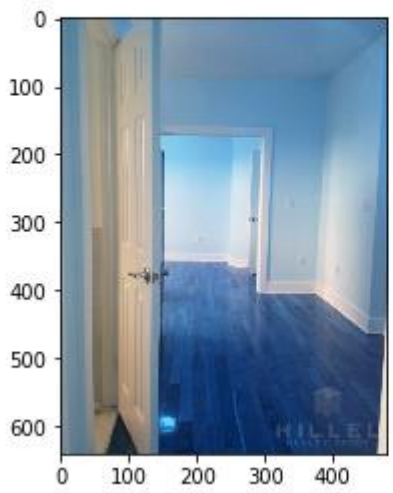

Рисунок 26 – Программный код и результат отображения фотографии

Дополнительно посчитаем количество фотографий в объявлениях. Реализуем подсчет количества фотографий с помощью переменной photos_dct типа dict (словарь). Для этого с помощью цикла for проводим итерационный перебор значений в столбце photos с одновременным подсчётом количества фотографий. Программный код показан на рисунке 27.

```
In [38]: photos_dct = dict()
         for photos in df['photos']:
             key = len(photos)
             if photos_dct.get(key, None):
                 photos_dct[key] += 1
             else:
                 photos_dct[key] = 1
```

Рисунок 27 – Создание словаря photos_dct

С помощью такого словаря можно смотреть, сколько объявлений имеют указанное количество фотографий. Например, можно вывести

количество объявлений, которое вовсе не содержат фотографий (рисунок 28). В данном случае таких объявлений 3615.

```
In [39]: photos_dct[0]
Out[39]: 3615
```

Рисунок 28 – Вывод количества объявлений без фотографий

При обучении классификатора популярности объявления мы будем использовать простой признак – количество фотографий в объявлении. Назовем это поле photos_qnt и произведем подсчет фотографий с помощью программного кода представленного на рисунке 29.

```
In [40]: df['photos_qnt'] = df.loc[:, 'photos'].apply(lambda x: len(x))
```

Рисунок 29 – Вывод поля photos_qnt (количество фотографий в объявлении)

Таким образом, был проведен предварительный анализ данных об объявлениях, на основе которого в исходной выборке данных были созданы дополнительные поля.

3.2 Программный код для настройки работы программного модуля

Выберем признаки данных, которые будут использованы при обучении классификатора популярности объявлений. Будем использовать следующие признаки:

- bathrooms – количество ванных комнат;
- bedrooms – количество спален;

- price – цена аренды;
- latitude – широта;
- longitude – долгота;
- created_year – год создания;
- created_month – месяц создания;
- created_day – день создания;
- photos_qnt – количество фотографий у объявления;
- features_qnt – количество символов текстового описания особенностей жилья;
- description_qnt – количество символов текстового описания жилья;
- tr_sparse – результат токенизации описания особенностей жилья.

Извлечем эти признаки из переменной df типа dataframe, а также исключим из рассмотрения объявления с аномальной ценой (индексы которых хранятся в переменной outliers_indexes). Результат этих действий сохраним в переменной X_train. Также, для удобства выведем размерность получившейся таблицы на экран. Программный код для выполнения этого этапа показан на рисунке 30.

Количество столбцов в переменной x_train равно 211. Значительный вклад в это количество внесла матрица tr_sparse – в ней содержится 200 столбцов (результат токенизации текстового описания особенностей жилья).

Выберем признаки для обучающей выборки

```
In [41]: feature_cols = ['bathrooms', 'bedrooms', 'price', 'latitude', 'longitude',
                         'created_year', 'created_month', 'created_day',
                         'photos_qnt', 'features_qnt', 'description_qnt']

In [42]: X_train = sparse.hstack([df.drop(outliers_indexes, axis=0)[feature_cols], tr_spa
         rse]).tocsr()  # Compressed Sparse Row matrix

In [51]: X_train.shape

Out[51]: (49348, 211)
```

Рисунок 30 – Подготовка матрицы X_train входных параметров для обучения классификатора

Теперь преобразуем целевую переменную, для этого переведем значения из текстового формата в числовой. Уровень интереса high будем кодировать 0, уровень medium кодируем значением 1, а уровень low – 2. Для этого зададим словарь перехода target_num_map, и применим его к столбцу interest_level, не забыв исключить объявления с аномальной ценой (индексы которых хранятся в переменной outliers_indexes). В результате получим вектор y_train, в котором будут храниться закодированные значения целевой переменной для всех объявлений. Программный код этого этапа представлен на рисунке 31.

```
In [43]: target_num_map = {'high': 0, 'medium': 1, 'low': 2}
         y_train = df.drop(outliers_indexes, axis=0)['interest_level'].apply(lambda x: ta
         rget_num_map[x])
```

Рисунок 31 – Подготовка вектора y_train целевых значений

Переменные X_train и y_train образуют обучающую выборку (основанную на данных из файла train.json) предназначенную для построения классификатора популярности объявлений. Для того, чтобы оценить качество обучаемых классификаторов необходимо аналогичным образом подготовить тестовые данные X_test на основе информации из файла test.json. Для этого аналогичным образом с помощью метода json.load() открываем поток на чтение файла train.json и сохраняем данные в переменную test_data. Затем переводим данные к типу dataframe с использованием метода from_dict(), задаем индекс, сортируем по индексу и выводим первые несколько записей на экран. Программный код представлен на рисунке 32.

Тестовые данные

```
In [44]: with open('test.json') as f:
             test_data = json.load(f)

In [45]: df_test = pd.DataFrame.from_dict(test_data)
         df_test.index = df_test.index.astype(np.int)
         df_test = df_test.sort_index()

In [46]: df_test.head()
```

Out[46]:

	bathrooms	bedrooms	building_id	created	description	display_addres
0	1.0	1	79780be1514f645d7e6be99a3de696c5	2016-06-11 05:29:41	Large with awesome terrace--accessible via bed...	Suffolk Stre
1	1.0	2	0	2016-06-24 06:36:34	Prime Soho - between Bleecker and Houston - Ne...	Thompson Stre
2	1.0	0	0	2016-06-17 01:23:39	Spacious studio in Prime Location. Cleanbuildi...	Sullivan Stre
3	1.0	2	f9c826104b91d868e69bd25746448c0c	2016-06-21 05:06:02	For immediate access call Bryan. Bo...	Jones Stre
5	1.0	1	81062936e12ee5fa6cd2b965698e17d5	2016-06-16 07:24:27	Beautiful TRUE 1 bedroom in a luxury building ...	Exchange Plac

Рисунок 32 – Загрузка тестовых данных

Аналогичным образом проведем предобработку имеющихся признаков, приведя их к следующему списку: bathrooms, bedrooms, price, latitude, longitude, created_year, created_month, created_day, photos_qnt, features_qnt, description_qnt, test_sparse. Программный код данного шага представлен на рисунке 33.

Выделим признаки из тестовой выборки аналогично обучающей выборке

```
In [47]: df_test['features_qnt'] = df_test.loc[:, 'features'].apply(lambda x: len(x))
         df_test['description_qnt'] = df_test.loc[:, 'description'].apply(lambda x: len
         (x))
         df_test['photos_qnt'] = df_test.loc[:, 'photos'].apply(lambda x: len(x))

In [48]: df_test['created'] = pd.to_datetime(df_test['created'])
         df_test['timestamp'] = df_test['created'].dt.date

         df_test['created_year'] = np.array([dt.year for dt in df_test['timestamp'].value
         s])
         df_test['created_month'] = np.array([dt.month for dt in df_test['timestamp'].val
         ues])
         df_test['created_day'] = np.array([dt.day for dt in df_test['timestamp'].value
         s])

In [49]: df_test['features_str'] = df_test['features'].apply(lambda x: ' '.join(['_'.join
         (i.split(' ')) for i in x]))
         test_sparse = tfidf.transform(df_test['features_str'])

In [50]: X_test = sparse.hstack([df_test[feature_cols], test_sparse]).tocsr()

In [52]: X_test.shape

Out[52]: (74659, 211)
```

Рисунок 33 – Загрузка тестовых данных и предобоаботка

Таким образом, мы получаем тестовые данные X_test полностью повторяющие по структуре тренировочные данные.

3.3 Тестирование и сравнение результатов работы алгоритмов

Для классификации объявлений по популярности мы будем обучать классификаторы, описанные в предыдущем разделе, именно:

- дерево классификации (Decision Tree Classifier),
- случайный лес (Random Forest Classifier),
- градиентный бустинг (Gradient Boosting Classifier),
- XGBoost.

Реализация дерева классификации импортирована из библиотеки sklearn. Так как мы решаем задачу обучения классификационной модели, то

будем пользоваться методом DecisionTreeClassifier(). Для того, чтобы определить классификатор зададим переменную clf_decision_tree. Для обучения классификатора необходимо вызвать метод fit(), передав на вход тренировочные данные (X_train и y_train). Программный код показан на рисунке 34.

Рисунок 34 – Код для обучения классификатора Decision Tree Classifier

Аналогичным образом импортируем из библиотеки sklearn, и обучаем классификатор на основе случайного леса (используя при этом те же данные). Программный код показан на рисунке 35.

Рисунок 35 – Код для обучения классификатора Random Forest Classifier

Аналогичным образом импортируем из библиотеки sklearn, и обучаем классификатор на основе градиентного бустинга (рисунок 36). Здесь необходимо задать дополнительный параметр – глубину градиентного бустинга (max_depth=6). Это параметр выбирается опытным путем.

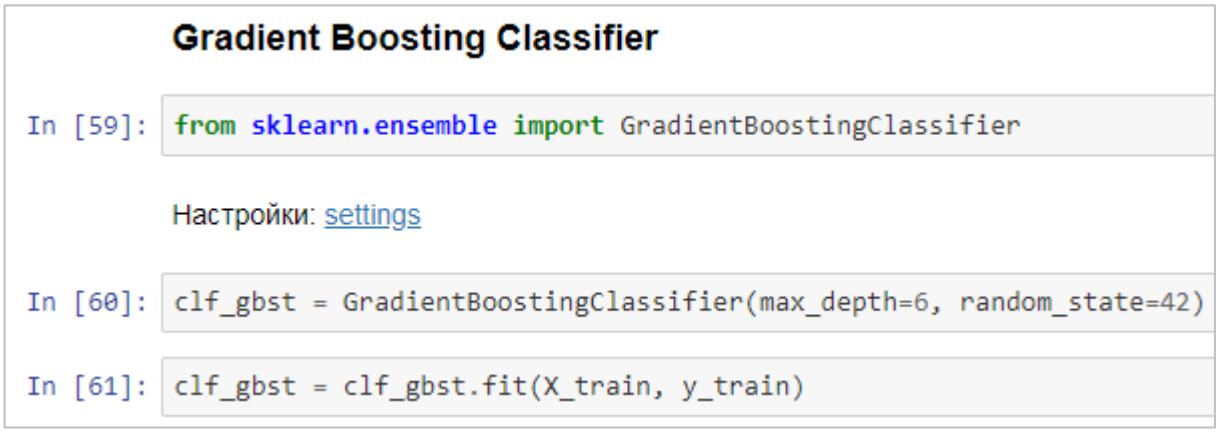

Рисунок 36 – Код для обучения классификатора Gradient Boosting Classifier

Реализация XGBoost импортирована из одноименной библиотеки xgboost. Так как данная библиотека по методам работы с данными отличается от sklearn, то аналогичным образом данные подать не получится. Данные в формате dataframe данная библиотека не поддерживает, поэтому необходимо преобразовать исходные данные с помощью метода DMatrix().

Также необходимо задать параметры обучения классификационной модели, в именно max_depth – максимальную глубину, eta – параметр критерия остановки, objective – целевая функция, nthread – количество потоков для параллелизации вычислений, eval_metric – метрика для посчета потерь, num_class – количество классов, num_rounds – количество итераций.

Запуск обучения классификационной модели XGBoost осуществляется с помощью метода train().

Программный код для импортирования модели XGBoost, задания параметров и обучения модели показан на рисунке 37.

XGBoost

In [62]:
```
import xgboost as xgb
```

Настройки: settings

In [63]:
```
dtrain = xgb.DMatrix(X_train, label=y_train)
dtest = xgb.DMatrix(X_test)
```

In [64]:
```
param = {'max_depth': 6, 'eta': 0.1, 'objective': 'multi:softprob'}
param['nthread'] = 4
param['eval_metric'] = 'mlogloss'
param['num_class'] = 3
```

In [65]:
```
num_rounds = 400
clf_xgb = xgb.train(param, dtrain, num_rounds)
```

Рисунок 37 – Код для обучения классификатора XGBoost

Теперь чтобы протестировать точность работы всех четырех полученных классификаторов воспользуемся методами predict() и predict_proba() предав в качестве входного параметра тестовые данные. Сохраним результаты работы классификаторов в отдельные переменные для дальнейшего анализа. Программный код, для выполнения данного шага показан на рисунке 38.

Предсказание на тестовом датасете

In [66]:
```
y_test_decision_tree = clf_decision_tree.predict_proba(X_test)
```

In [67]:
```
y_test_random_forest = clf_random_forest.predict_proba(X_test)
```

In [68]:
```
y_test_gbst = clf_gbst.predict_proba(X_test)
```

In [69]:
```
y_test_xgb = clf_xgb.predict(dtest)
```

Рисунок 38 – Получение результатов работы 4 классификаторов на тестовой выборке данных

Для того чтобы проверить точность работы классификаторов с использованием сети kaggle, необходимо подготовить csv файлы. Эти csv файлы должны содержать результаты работы классификаторов на тестовых данных. Впоследствии csv файлы загружаются через веб-форму на сайте kaggle.com и в качестве ответа возвращаются показатели точности работы классификаторов. Код создания csv файлов с результатами работы классификаторов показан на рисунке 39.

Results

In [70]:
```
predictions_dct = {'decision_tree': y_test_decision_tree, 'random_forest': y_tes
t_random_forest,
                   'gbst': y_test_gbst, 'xgb': y_test_xgb}
```

In [71]:
```
for method, y_test in predictions_dct.items():
    result = pd.DataFrame({'listing_id': df_test['listing_id'].tolist(),
                           'high': list(y_test[:, 0]),
                           'medium': list(y_test[:, 1]),
                           'low': list(y_test[:, 2])})
    result = result.set_index('listing_id')
    print(method)
    print(result.head())
    result.to_csv(method + '.csv', sep=',')
```

Рисунок 39 – Код создания csv файлов с результатами работы 4 классификаторов

Содержимое каждого из 4 файлов (decision_tree.csv, random_forest.csv, gbst.csv и xgb.csv) показано на рисунках 40-43.

```
decision_tree
             high  low  medium
listing_id
7142618       0.0  0.0     1.0
7210040       0.0  1.0     0.0
7174566       0.0  1.0     0.0
7191391       0.0  1.0     0.0
7171695       0.0  1.0     0.0
```

Рисунок 40 – Первые несколько строк файла decision_tree.csv с результатами работы классификатора

random_forest

listing_id	high	low	medium
7142618	0.04	0.67	0.29
7210040	0.07	0.80	0.13
7174566	0.00	0.99	0.01
7191391	0.06	0.43	0.51
7171695	0.05	0.72	0.23

Рисунок 41 – Первые несколько строк файла random_forest.csv с результатами работы классификатора

gbst

listing_id	high	low	medium
7142618	0.062139	0.596291	0.341570
7210040	0.138646	0.631401	0.229953
7174566	0.014070	0.945877	0.040053
7191391	0.259607	0.231533	0.508859
7171695	0.010211	0.855011	0.134778

Рисунок 42 – Первые несколько строк файла gbst.csv с результатами работы классификатора

xgb

listing_id	high	low	medium
7142618	0.053877	0.492399	0.453724
7210040	0.098385	0.719238	0.182377
7174566	0.004802	0.968552	0.026646
7191391	0.286351	0.238257	0.475393
7171695	0.022595	0.805499	0.171906

Рисунок 43 – Первые несколько строк файла xgb.csv с результатами работы классификатора

Полученные csv файлы были загружены через веб-форму сайта kaggle (рисунок 44). В результате были получены следующие показатели точности работы 4 классификаторов:

- Decision Tree Classifier Scores: 11.76794;
- Random Forest Classifier Scores: 0.66832;

- Gradient Boosting Classifier Scores: 0.59352;
- XGBoost Classifier Scores: 0.58381.

Графическая интерпретация полученных результатов в виде столбчатой диаграммы показана на рисунке 45.

Также стоит отметить, что по правилам kaggle, чем меньше значение score набрано моделью, тем точнее она работает.

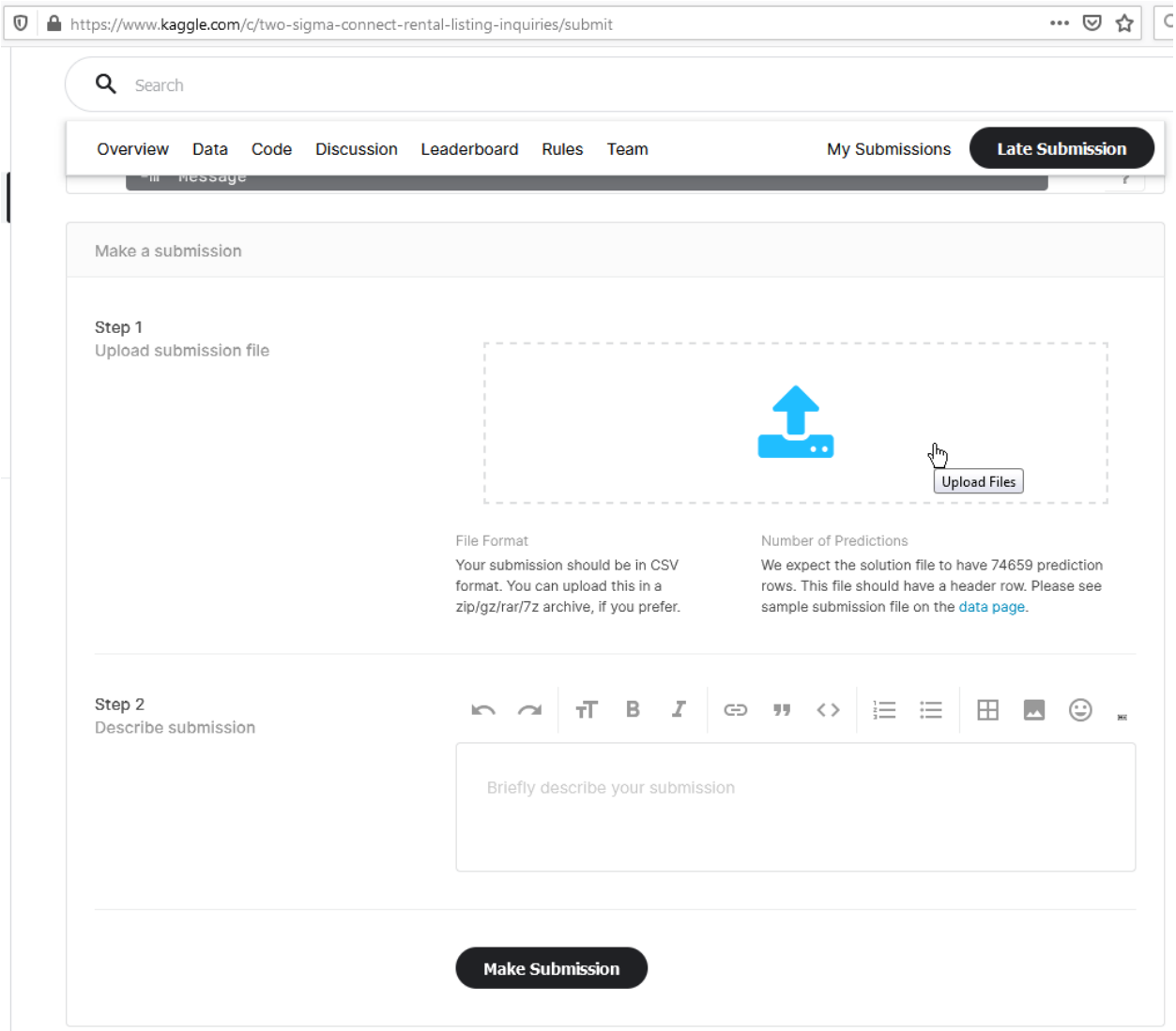

Рисунок 44 – Веб-форма для загрузки csv файлов и получение численной оценки работы классификаторов

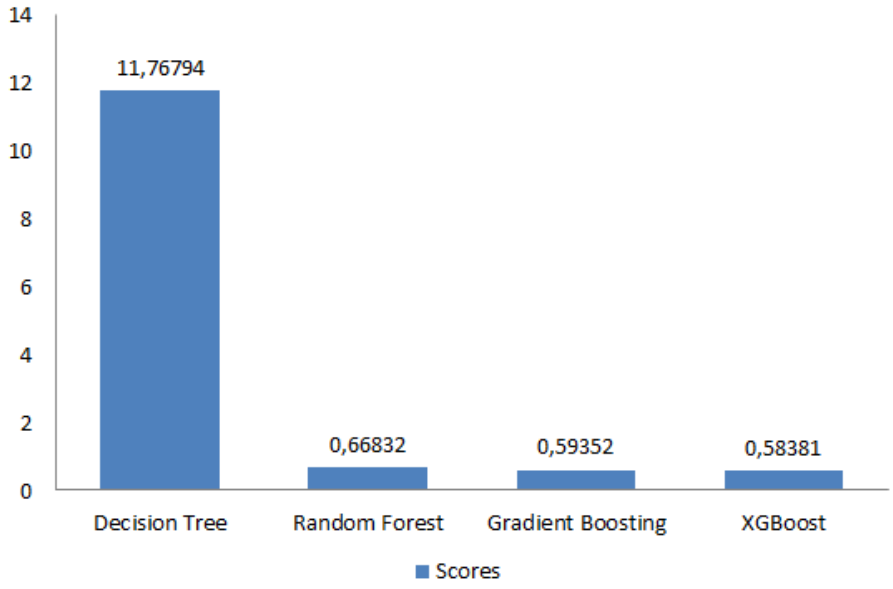

Рисунок 45 – Результаты оценки точности классификаторов, выданных системой kaggle (чем меньше значение Score, тем лучше)

Таким образом, наиболее точной моделью классификации объявлений об аренде жилья является модель, основанная на XGBoost. Худшие результаты на представленных данных показала модель, основанная на Decision Tree.

3.4 Интерфейс программного обеспечения

Пользователь сайта об аренде недвижимости, формируя запрос через веб-форму, получают список объявлений, отсортированный по дате создания объявлений. Для того чтобы повысить вероятность аренды жилья через сайт компании (а, следовательно, и прибыль компании) предлагается сортировать объявления по прогнозируемой популярности.

Для обеспечения удобства работы пользователя с аналитическим программным модулем анализа популярности объявления был разработан графический интерфейс. Логически интерфейс поделен на две части:

- блок «Пакетная обработка объявлений», в котором предложено выбрать файл в формате json с исходными данными нескольких

объявлений, для которых необходимо спрогнозировать популярность. Данный блок содержит в себе кнопку «Обзор», открывающую проводник для выбора файла с данными объявлений. Результаты анализа буду сохранены в папку с проектом файл под именем «output.json».

- блок «Обработка одного объявления», в котором можно с помощью элементов управления последовательно задать информацию объявления, для которого требуется спрогнозировать популярность (рисунок 46).

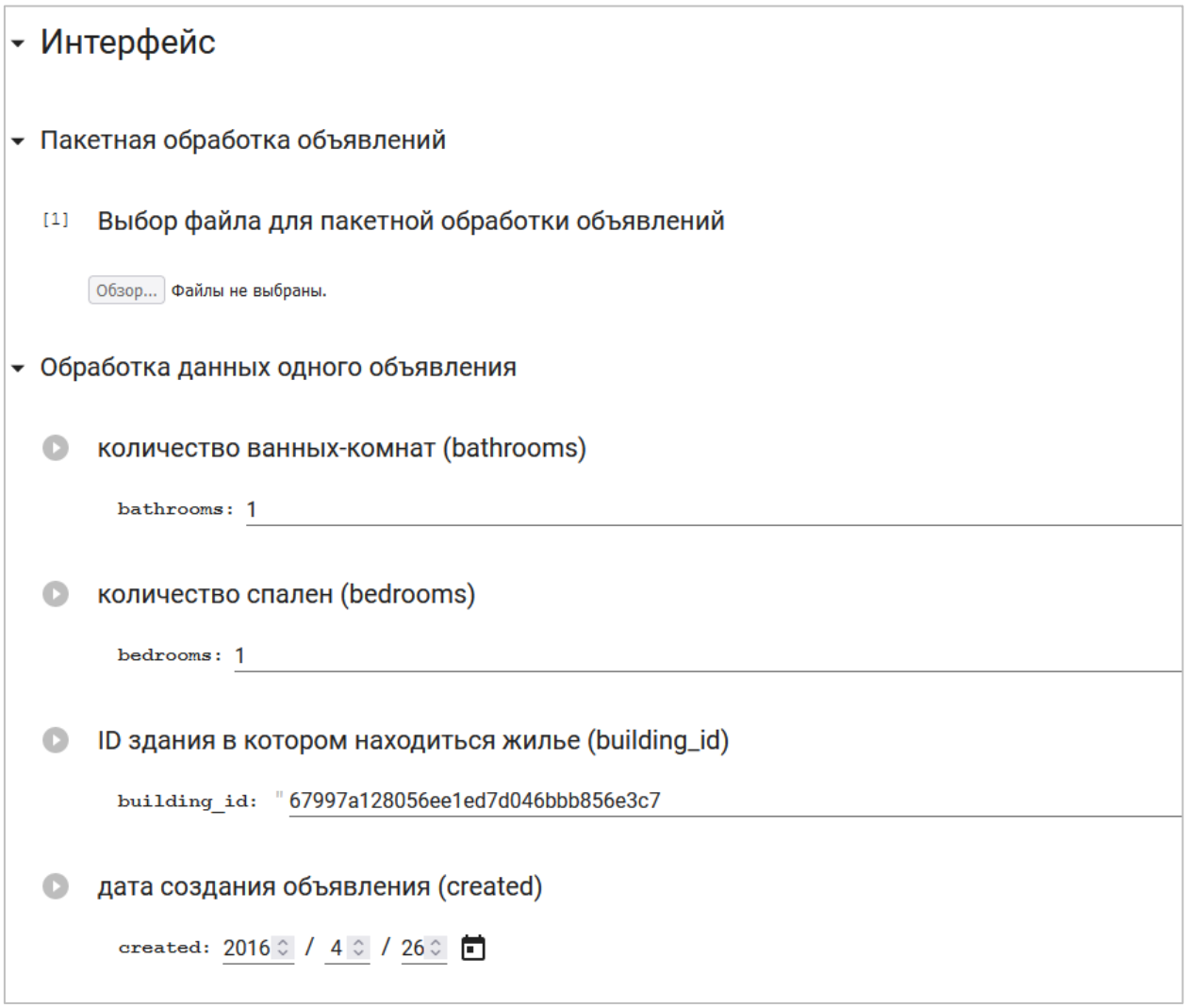

Рисунок 46 – Интерфейс аналитического программного модуля для прогнозирования популярности объявлений

В нижней части блока «Обработка одного объявления» расположена кнопка «Анализ», при нажатии на которую проводиться анализ одиночного объявления с заданными пользователем параметрами. Определённый класс популярности объявления выводится в виде текста под кнопкой (рисунок 47).

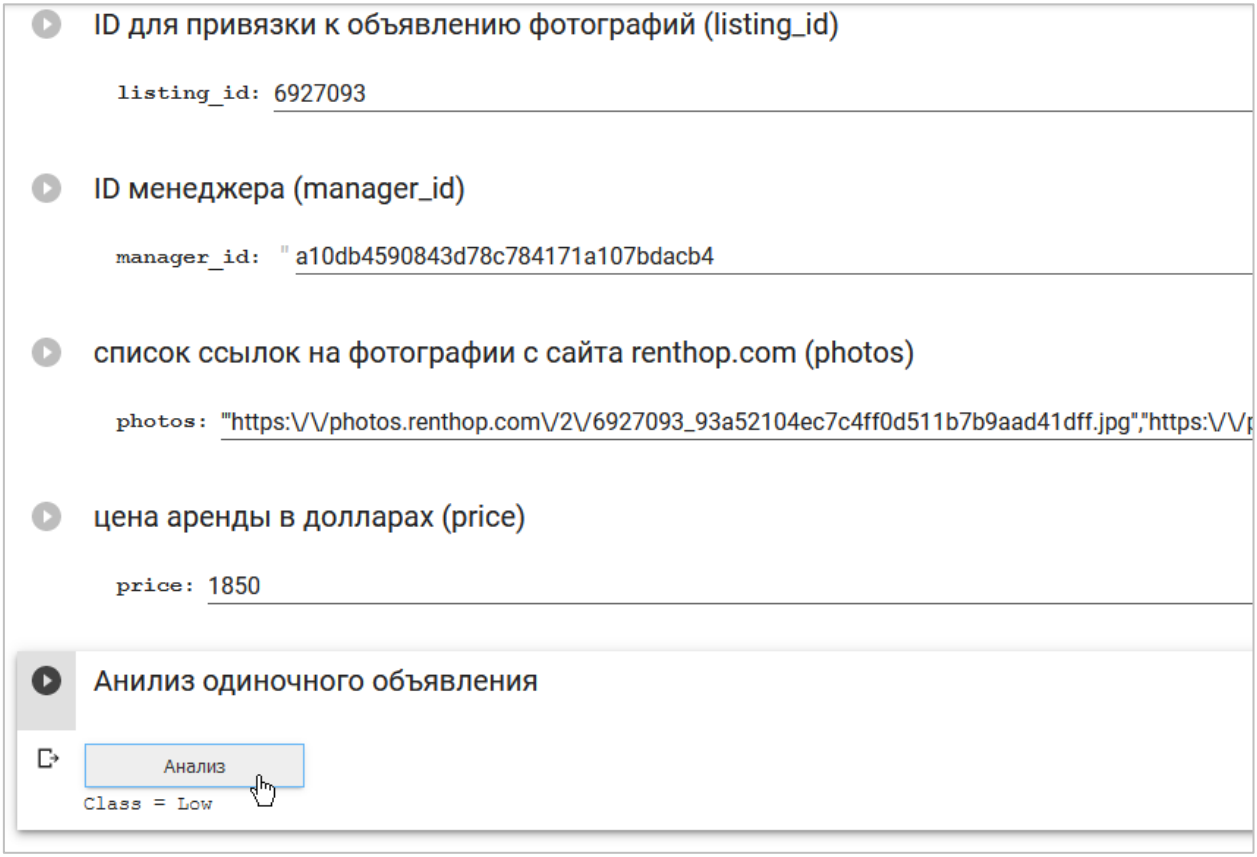

Рисунок 47 – Интерфейс блока для анализа одиночного объявления

Тестирование программного обеспечения проводилось опытным путем. Результат работы программного модуля на данных одного объявления показан на рисунке 47.

Выводы по третьей главе

На языке Python разработано программное обеспечение, реализующее предсказания класса популярности недвижимости (high, medium, low) на основе разнородного содержания объявления о квартире: текстовом описании, фотографиях, количестве спален, цене и т.д. Данные поступают с сайта по аренде квартир (renthop.com).

Разработанное программное обеспечение обеспечивает построение классификаторов в соответствии с алгоритмами Decision Tree, Random Forest, Gradient Boosting и XGBoost.

Проведены вычислительные эксперименты по оценки точности работы 4 классификаторов. Лучшие результаты показал классификатор на основе XGBoost, а худшие результаты – на основе Decision Tree.

Заключение

Целью работы являлась разработка программного модуля обучения классификатора для прогнозирования популярности объявлений.

Достижению поставленной цели способствовало успешное решение основных задач исследования. В частности, был произведен анализ бизнес-процессов компании renthop, предоставляющей информационные услуги по поиску объявлений об аренде жилья. Для усовершенствования исследуемого бизнес-процесса, связанного с выводом результатов поиска объявлений пользователю, предложено разработать и внедрить программное обеспечение для прогнозирования популярности объявлений.

Прогнозирование популярности объявлений о сдаче квартир в аренду представлено в виде задачи классификации. Для прогнозирования обучен классификатор, способный на основе входных параметров объявлений, определять класс его популярности.

Рассмотрены алгоритмы машинного обучения, способные решать задачи классификации и основанные на использовании деревьев принятия решений. На основе их различий и особенностей составлена сводная таблица.

На языке Python разработано программное обеспечение, реализующее предсказания класса популярности недвижимости (high, medium, low) на основе разнородного содержания объявления о квартире: текстовом описании, фотографиях, количестве спален, цене и т.д. Данные поступают с сайта по аренде квартир (renthop.com). Разработанное программное обеспечение обеспечивает построение классификаторов в соответствии с алгоритмами Decision Tree, Random Forest, Gradient Boosting и XGBoost.

Проведены вычислительные эксперименты по оценки точности работы 4 классификаторов. Лучшие результаты показал классификатор на основе XGBoost, а худшие результаты – на основе Decision Tree.

Список используемой литературы

1. Александров, Д. В. Инструментальные средства информационного менеджмента : CASE-технологии и распределенные информационные системы : учебное пособие / Д. В. Александров. – М. : Финансы и статистика, 2011 – 224 с. – Текст : непосредственный.

2. Аусабаев, Д.М. Использование машинного обучения в поддержке принятия решений / Д.М. Аусабаев, О.П. Волобуев // Прикладная математика и информатика: современные исследования в области естественных и технических наук – материалы III научно-практической всероссийской конференции (школы-семинара) молодых ученых. Тольятти, 24–25 апреля 2017 года. – Издатель Качалин Александр Васильевич, 2017. – с. 43-47. – Текст : непосредственный.

3. Власов, А.В. Машинное обучение применительно к задаче классификации семян зерновых культур в видеопотоке / А.В. Власов, А.С. Федеев // Молодежь и современные информационные технологии – сборник трудов XIV Международной научно-практической конференции студентов, аспирантов и молодых учёных, 07–11 ноября 2016. – Национальный исследовательский Томский политехнический университет (Томск), 2016. – с. 133-135. – Текст : непосредственный.

4. Жуков, Д.А. Формирование контрольных выборок при технической диагностике объекта с применением машинного обучения / Д.А. Жуков, А.С. Хорева, Ю.Е. Кувайскова, В.Н. Клячкин // Математические методы и модели: теория, приложения и роль в образовании – международная научно-техническая конференция : сборник научных трудов, 28–30 апреля 2016 года. – Ульяновский государственный технический университет (Ульяновск), 2016. – с. 44-48. – Текст : непосредственный.

5. Иванников Ю.Ю. Применение методов машинного обучения для выявления бот-трафика среди запросов к веб-приложению / Ю.Ю. Иванников, Е.Ю. Митрофанова // Сборник студенческих научных работ

факультета компьютерных наук ВГУ, Факультет компьютерных наук, 2017. – ФГБОУ ВО «Воронежский государственный университет», 2017. – с. 119-123. – Текст : непосредственный.

6. Клячин В.Н. Использование агрегированных классификаторов при технической диагностике на базе машинного обучения / В.Н. Клячин, Ю.Е. Кувайскова, Д.А. Жуков // Информационные технологии и нанотехнологии (ИТНТ-2017) – сборник трудов III международной конференции и молодежной школы. Самарский национальный исследовательский университет имени академика С.П. Королева. 2017. – Предприятие «Новая техника» (Самара), 2017. – с. 1770-1773. – Текст : непосредственный.

7. Кононова, Н.В. Исследование подсистемы контентной фильтрации с использованием методов машинного обучения / Н.В. Кононова, Ю.А. Андрусенко, Т.А. Самокаева // Студенческая наука для развития информационного общества – сборник материалов VI Всероссийской научно-технической конференции. 22–26 мая 2017. – Северо-Кавказский федеральный университет (Ставрополь), 2017. – с. 268-270. – Текст : непосредственный.

8. Мелдебай, М.А. Анализ мнений покупателей на основе машинного обучения / М.А. Мелдебай, А.К. Сарбасова // Прикладная математика и информатика: современные исследования в области естественных и технических наук – материалы III научно-практической всероссийской конференции (школы-семинара) молодых ученых. 24–25 апреля 2017 года. – Издатель Качалин Александр Васильевич, 2017. – с. 360-363. – Текст : непосредственный.

9. Наумов, Д.П. Регулятор САР на основе машинного обучения / Д.П. Наумов, Д.П. Стариков // Информационные технологии в управлении, автоматизации и мехатронике – сборник научных трудов Международной научно-технической конференции. 06–07 апреля 2017 года. – ЗАО

«Университетская книга» (Курск), 2017. – с. 106-114. – Текст : непосредственный.

10. Осколков, В.М. Использование метода машинного обучения для повышения продуктивности на предприятии / В.М. Осколков, Н.И. Шаханов, И.А. Варфоломеев, О.В. Юдина, Е.В. Ершов // Автоматизация и энергосбережение машиностроительного и металлургического производств, технология и надежность машин, приборов и оборудования – материалы XII Международной научно-технической конференции, 21 марта 2017. – Вологодский государственный университет (Вологда), 2017. – с. 177-180. – Текст : непосредственный.

11. Осколков, В.М. Применение параллельных вычислений для прогнозирования на основе алгоритма машинного обучения Random Forest / В.М. Осколков, Н.И. Шаханов, И.А. Варфоломеев, О.В. Юдина, Л.Н. Виноградова, Е.В. Ершов // Сборник трудов конференции Оптико-электронные приборы и устройства в системах распознавания образов, обработки изображений и символьной информации. Распознавание, Курск, 16–19 мая 2017 года. – Юго-Западный государственный университет (Курск), 2017. – с. 267-269. – Текст : непосредственный.

12. Соловьев, А.Ю. Применение машинного обучения для прогнозирования неблагоприятных исходов в ургентной хирургии / Соловьев А.Ю., Берегов М.М., Вахеева Ю.М., Баутин А.Н., Гусев А.В. // Медико-биологические, клинические и социальные вопросы здоровья и патологии человека – материалы III Всероссийской образовательно-научной конференции студентов и молодых ученых с международным участием в рамках XIII областного фестиваля "Молодые ученые - развитию Ивановской области". 2017. – Ивановская государственная медицинская академия (Иваново), 2017. – с. 129-130. – Текст : непосредственный.

13. Ткач, Т.Ч. Машинное обучение и обработка больших данных - обучение в основной и средней школе / Т.Ч. Ткач // Актуальные проблемы методики обучения информатике и математике в современной школе –

материалы международной научно-практической интернет-конференции. Московский педагогический государственный университет, Москва, 24 апреля 2020 года. – Московский педагогический государственный университет (Москва), 2020. – с. 217-223. – Текст : непосредственный.

14. Федотов, И.А. Применение технологий машинного обучения для прогнозирования ситуации на финансовых рынках / И.А. Федотов // Студенческая наука для развития информационного общества – сборник материалов VI Всероссийской научно-технической конференции. 22–26 мая 2017. – Северо-Кавказский федеральный университет (Ставрополь), 2017. – с. 361-363. – Текст : непосредственный.

15. Якимчук, А.А. Глубокое обучение как эффективный метод машинного обучения / А.А. Якимчук // Научное сообщество студентов XXI столетия. Технические науки – сборник статей по материалам XCII студенческой международной научно-практической конференции. 2020. – ООО “Сибирская академическая книга” (Новосибирск), 2020. – с. 40-43. – Текст : непосредственный.

16. Ярыгин, А.А. Актуальные вопросы машинного обучения с подкреплением интеллектуальных агентов в задачах принятия решений / А.А. Ярыгин // Автоматизация: проблемы, идеи, решения - сборник статей по итогам Международной научно-практической конференции 2017. – ООО "Агентство международных исследований", 2017. – с. 62-68. – Текст : непосредственный.

17. Bartczuk, Ł. A New Version of the Fuzzy-ID3 Algorithm / Łukasz Bartczuk, Danuta Rutkowska // International Conference on Artificial Intelligence and Soft Computing – 8th International Conference, Zakopane, Poland, June 25-29, 2006. Proceedings: Artificial Intelligence and Soft Computing – ICAISC 2006. – Springer-Verlag Berlin Heidelberg 2006. – pp. 1060-1070. – Текст : непосредственный.

18. Chang, J. Genetic Algorithm Based Fuzzy ID3 Algorithm [Text] / Jyh-Yeong Chang, Chien-Wen Cho, Su-Hwang Hsieh, Shi-Tsung Chen //

International Conference on Neural Information Processing – 11th International Conference, ICONIP 2004, Calcutta, India, November 22-25, 2004. Proceedings: Neural Information Processing. – Springer-Verlag Berlin Heidelberg 2004. – pp. 989-995. – Текст : непосредственный.

19. Jiang, S. A Combined Classification Algorithm Based on C4.5 and NB / ShengYi Jiang, Wen Yu // International Symposium on Intelligence Computation and Applications – Third International Symposium, ISICA 2008 Wuhan, China, December 19-21, 2008. Proceedings: Advances in Computation and Intelligence. – Springer-Verlag Berlin Heidelberg 2008. – pp. 350-359. – Текст : непосредственный.

20. Min, F. A Competition Strategy to Cost-Sensitive Decision Trees / Fan Min, William Zhu // International Conference on Rough Sets and Knowledge Technology – 7th International Conference, RSKT 2012, Chengdu, China, August 17-20, 2012. Proceedings: Rough Sets and Knowledge Technology. – Springer-Verlag Berlin Heidelberg 2012. – pp. 359-368. – Текст : непосредственный.

21. Zheng, Z. Scaling up the rule generation of C4.5 / Zijian Zheng // Pacific-Asia Conference on Knowledge Discovery and Data Mining – Second Pacific-Asia Conference, PAKDD-98 Melbourne, Australia, April 15–17, 1998. Proceedings: Research and Development in Knowledge Discovery and Data Mining. – Springer-Verlag Berlin Heidelberg 1998. – pp. 348-359. – Текст : непосредственный. – Текст : непосредственный.