МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ «МИСиС»

Институт ИТКН

Кафедра инженерной кибернетики

Направление подготовки: «01.03.04 Прикладная математика»

Квалификация: бакалавр

Группа: БПМ-17-1

ОТЧЕТ

ПО КУРСОВОЙ РАБОТЕ «ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ»

на тему: Нейронная сеть для предсказания стоимости подержанных автомобилей по их характеристикам

Студент (ы) _____/ Кравчук Никита Александрович /
подпись (ф.и.о. полностью)

Руководитель _____/ ст. преп. Кондыбаева А.Б. /
подпись должность, уч. степ. Фамилия И.О.

Оценка: _____

Дата защиты: _____

Москва 2020

Содержание

Введение

Тема данной курсовой работы – «Нейронная сеть для предсказания стоимости подержанных автомобилей по их характеристикам».

В настоящее время алгоритмы машинного обучения (ML) позволяют решать множество задач, включая задачи прогнозирования цен на товары. В данной работе рассмотрено обучение нейронной сети для предсказания цен на подержанные автомобили.

В результате выполнения курсовой работы реализовано приложение с графическим интерфейсом пользователя (GUI), позволяющее по техническим характеристикам и информации о состоянии автомобиля предсказывать его рыночную стоимость. Прогнозирование стоимости осуществляется при помощи многослойной нейронной сети. Также в целях сравнения с нейронной сетью были рассмотрены градиентный бустинг над решающими деревьями и случайный лес. Для каждого применённого метода проведена оценка качества его работы на тестовой выборке по трём различным метрикам.

Курсовая работа выполнена студентом группы БПМ-17-1 Кравчуком Н.А.

Постановка задачи

Реализовать программное обеспечение, позволяющее по указанным характеристикам подержанного автомобиля прогнозировать его рыночную стоимость при помощи многослойной нейронной сети и 2 методов машинного обучения: градиентного бустинга над решающими деревьями и случайного леса. Программное обеспечение должно быть реализовано в виде десктопного приложения с графическим интерфейсом пользователя.

Для решения постановленной задачи необходимо выполнить следующие шаги:

– найти датасет, содержащий описание подержанных автомобилей и их стоимость;

– проверить выбранный датасет на наличие выбросов и пропусков, выполнить предобработку исходных данных, определив способ заполнения недостающих данных, удалив выбросы и выполнив кодирование категориальных признаков;

– обучить нейронную сеть, градиентный бустинг над решающими деревьями и случайный лес на датасете, проверить качество их работы при помощи различных метрик и сохранить обученные модели в формате, предоставляемом библиотеками, для их дальнейшего использования в приложении;

– реализовать графический интерфейс пользователя;

– выполнить тестирование реализованного приложения.

Предметной областью в данной задаче является рынок подержанных автомобилей. В качестве исходных данных выступает датасет с портала Kaggle [1]. В данном датасете содержится такая информация об автомобилях, как производитель, модель, год выпуска, состояние, пробег, вид используемого топлива, объём двигателя, цвет, трансмиссия, тип привода, сегмент, к которому

принадлежит модель, и стоимость автомобиля в долларах. Примеры исходных данных представлены в таблице 1.

Таблица 1 – Пример исходных данных

Характеристика	Пример №1	Пример №2	Пример №3
Производитель	Hyundai	Ford	Opel
Модель	Tucson	Fiesta	Zafira
Год выпуска	2008	2009	2001
Состояние	С пробегом	С пробегом	Битый
Пробег, км	90981	176000	300000
Вид топлива	Бензин	Бензин	Дизель
Объём двигателя, $см^3$	1996	1300	2000
Цвет	Серебряный	Серебряный	Синий
Трансмиссия	Механика	Механика	Механика
Тип привода	Передний	Передний	Передний
Сегмент	J	B	M
Стоимость, доллары США	6893	5500	2000

Использованные средства разработки и системные требования

Разработка программного обеспечения выполнялась на языке Python. Обработка данных, обучение моделей и проверка качества их работы производились в среде Jupyter Notebook. На этапе предобработки данных и обучения моделей были использованы следующие библиотеки и фреймворки:

– библиотеки NumPy и pandas для работы с исходными данными;

– фреймворк PyTorch для реализации нейронной сети;

– библиотека scikit-learn для применения случайного леса;

– библиотека XGBoost для применения градиентного бустинга;

– библиотеки Matplotlib и seaborn для визуализации графиков;

– библиотека Joblib для сохранения моделей, предоставляемых scikit-learn.

Разработка интерфейса приложения производилась в IDE PyCharm. Графический интерфейс пользователя был реализован при помощи библиотеки PyQT.

В таблице 2 представлены системные требования для работы приложения.

Таблица 2 – Системные требования

Платформа	MacOS, Linux, Windows
CPU	Intel Core i3-6100U 2.30 ГГц
Оперативная память	4 Гб
Свободное дисковое пространство	450 Мб

Научный аппарат

Градиентный бустинг над решающими деревьями

Идея бустинг-подхода заключается в комбинации слабых (с невысокой обобщающей способностью) функций, которые строятся в ходе итеративного процесса, где на каждом шаге новая модель обучается с использованием данных об ошибках предыдущих. Результирующая функция представляет собой линейную комбинацию базовых, слабых моделей. В градиентном бустинге над деревьями в качестве базовых моделей используются решающие деревья.

В процессе обучения градиентного бустинга на каждом шаге выполняется добавление в ансамбль дерева, которое улучшает ответы предыдущих моделей. Пусть на шаге t имеются функции $f_i, 1 \leq i \leq t-1$, которые определяются структурой деревьев решений. Обозначим через $\hat{y}_i^{(t)}$ предсказанный градиентным бустингом ответ на каждом из n объектов обучающей выборки x_i на шаге t. Тогда $\hat{y}_i^{(t)}$ вычисляется по формуле

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i). \quad (1)$$

Добавляемая на шаге t модель f_t должна минимизировать эмпирический риск obj^t:

$$obj^t = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \sum_{i=1}^{t} \Omega(f_t), \quad (2)$$

где n – количество объектов в обучающей выборке;

l – функция потерь;

$y_i, 1 \leq i \leq n$ – ответы для объектов x_i;

Ω – функция, сопоставляющая модели f её сложность.

Начальное значение $\hat{y}_i^{(0)}$ принимается равным 0.

Случайный лес

Случайный лес – алгоритм машинного обучения, заключающийся в использовании комитета (ансамбля) решающих деревьев. Алгоритм сочетает в себе две основные идеи: метод бэггинга Бреймана и метод случайных подпространств. Алгоритм применяется для задач классификации, регрессии и кластеризации. Основная идея заключается в использовании большого ансамбля решающих деревьев, каждое из которых само по себе даёт очень невысокое качество классификации, но за счёт их большого количества результат получается хорошим.

Алгоритм обучения каждой модели, входящей в состав случайного леса, состоит из следующих шагов:

– выбирается подвыборка обучающей выборки размера samplesize, и по ней строится дерево (для каждого дерева – своя подвыборка);

– для построения каждого расщепления в дереве просматривают max_features случайных признаков (для каждого нового расщепления – свои случайные признаки);

– выбирают наилучшие признак и расщепление по нему (по заранее заданному критерию); дерево строится, как правило, до исчерпания выборки (пока в листьях не останутся представители только одного класса).

В случае задачи регрессии при прогнозе в качестве ответа случайного леса принимают среднее значение ответов всех моделей, входящих в состав ансамбля.

Многослойный перцептрон

Многослойный перцептрон – это алгоритм обучения с учителем, аппроксимирующий функцию $f: R^m \rightarrow R^o$ путём обучения на выборке. Значение m представляет собой размерность входного вектора, а o – размерность выходного вектора. Многослойный перцептрон может использоваться как в задачах классификации, так и в задачах регрессии. На рисунке 1 изображён пример многослойного перцептрона с одним скрытым слоем и выходным вектором размера 1.

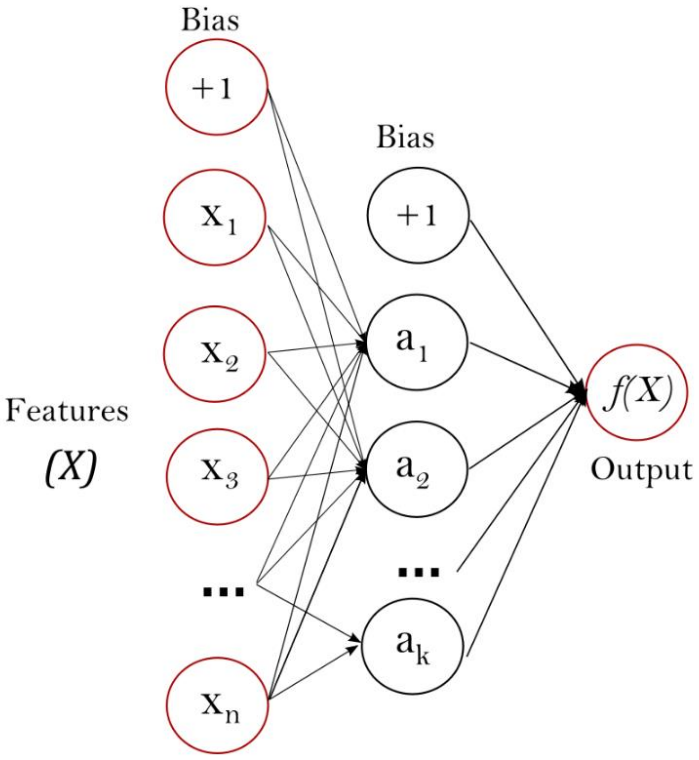

Рисунок 1 – Многослойный перцептрон

Первый входной слой состоит из множества нейронов $\{x_i\}_{i=1}^m$, которые представляют собой признаки объектов. Каждый нейрон в скрытом слое преобразует значения с предыдущего слоя в следующем виде:

$$w_1 x_1 + w_2 x_2 + \cdots + w_m x_m, \quad (3)$$

где $\{w_i\}_{i=1}^{m}$ – веса скрытого слоя.

К полученной сумме применяют функцию активации $g: R \to R$. Аналогичным образом выходной слой принимает значения с последнего скрытого слоя и преобразует их в выходное значение.

Обучение моделей и оценка качества их работы

Перед обучением моделей была проведена предобработка исходных данных. Были удалены записи с объёмом двигателя, превышающим 10 л, и с пробегом больше 1000000 км. Также были обнаружены пропуски данных в столбцах с типом привода, сегментом и объёмом двигателя автомобиля. Для заполнения пропусков в данных для каждой модели были определены самые частые параметры. В случае, если для всех автомобилей некоторой марки отсутствовала информация по одному из признаков, пропущенные значения для объёма двигателя было решено заполнить средним значением по всей выборке, а пропуски для типа привода и сегмента – самым встречающимся значением по всей выборке. После было выполнено кодирование категориальных признаков с помощью подхода One-Hot-Encoding. Перед использованием данных для обучения нейронной сети также была проведена их нормализация.

После обработки данных был проведён поиск гиперпараметров для градиентного бустинга и случайного леса, минимизирующих их ошибку на кросс-валидации. В результате для градиентного бустинга был выбрана скорость обучения 0,01, максимальная глубина решающих деревьев, равная 10, и количество моделей в ансамбле, равное 4000. Количество базовых моделей в случайном лесе было принято равным 100.

Реализованная нейронная сеть имеет 1 скрытый слой с 500 нейронами и выходной слой с одним нейроном. За каждым слоем нейронной сети следует функция активации ReLU. Нейронная сеть была обучена с количеством эпох, равным 50, скоростью обучения 0,01 и размером батча 64. На рисунке 2 изображен график функции потерь MSE на обучающей и валидационной выборке для каждой из 50 эпох.

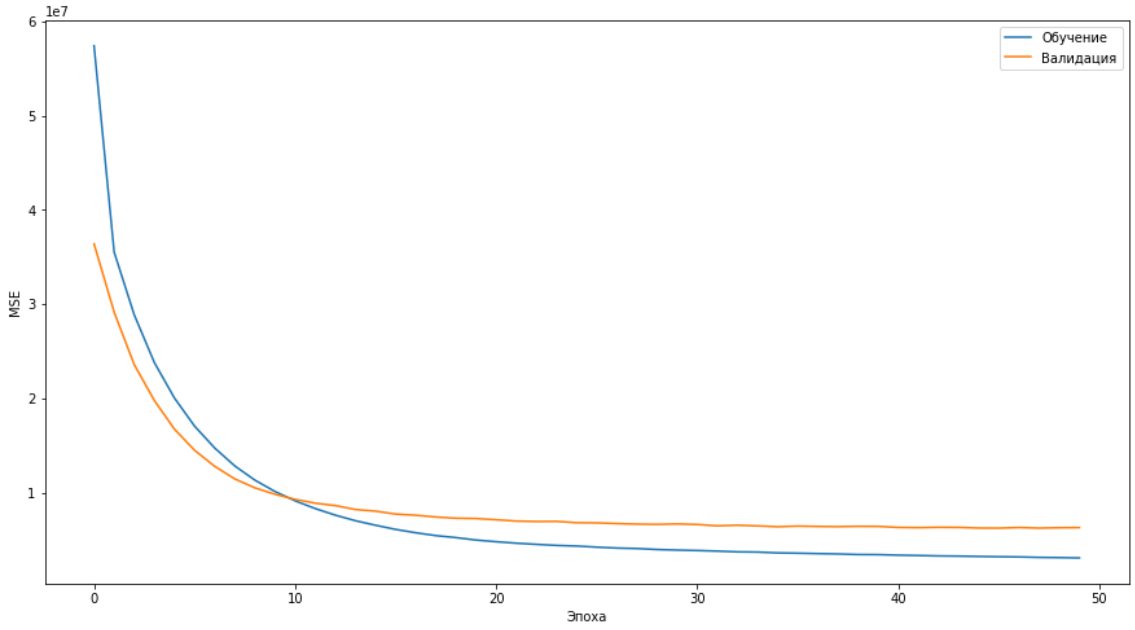

Рисунок 2 – Значения функции потерь при обучении нейронной сети

Для сравнения качества работы моделей машинного обучения, применённых для задачи прогнозирования стоимости подержанных автомобилей, была произведена их оценка по 3 метрикам: MAE (mean absolute error), MAPE (mean absolute percentage error) и MedAPE (median absolute percentage error). MAE и MAPE вычисляются по формулам

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|, \tag{4}$$

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|, \tag{5}$$

где n – размер обучающей выборки;

y_i – истинные ответы;

$\hat{y}_i$ – ответы модели.

Значение метрики MedAPE равно медиане значений $100 * \left|\frac{y_i - \hat{y}_i}{y_i}\right|$.

Проверка качества работы случайного леса и градиентного бустинга над решающими деревьями проводилась при помощи кросс-валидации с 5 разбиениями выборки. Оценка качества работы нейронной сети осуществлялась на 1 разбиении выборки на обучающую и тестовую. В результате были получены значения метрик, представленные в таблице 3.

Таблица 3 – Оценка качества обученных моделей

Метод	MAE	MAPE	MedAPE
Градиентный бустинг над деревьями	1030,88	23,6	12,01
Случайный лес	1066,65	23,42	12,07
Нейронная сеть	1000,35	22,15	11,61

Разработанное программное обеспечение

Общие сведения

Разработанное программное обеспечение служит для предсказания стоимости подержанных автомобилей. После запуска приложения пользователю необходимо указать директорию, в которой хранятся файлы с сохранёнными моделями машинного обучения и списком производителей, моделей автомобилей и возможных значений сегментов.

Для получения прогноза стоимости автомобиля пользователю необходимо ввести с помощью элементов графического интерфейса информацию об автомобиле. После этого данные передаются загруженным моделям, которые выполняют прогнозирование стоимости автомобиля. После завершения работы моделей предсказанные значения стоимости появляются в соответствующем окне.

Для предотвращения зависания пользовательского интерфейса загрузка сохранённых моделей и списка возможных значений для производителей, моделей и классов автомобилей, а также обработка входных данных и прогнозирование стоимости выполняются в отдельном потоке.

Список классов, разработанных для работы приложения

При реализации приложения был применён подход объектно-ориентированного программирования. В таблице 4 представлено описание классов, которые были разработаны для работы приложения.

Таблица 4 – Реализованные классы

Класс	Описание
Analyzer	Класс, выполняющий загрузку служебных данных и прогнозирование стоимости автомобиля при помощи моделей машинного обучения
Car	Класс для хранения информации об автомобиле
MainWindow	Класс, описывающий логику работы главного окна приложения
RegressionNet	Нейронная сеть для прогнозирования стоимости автомобилей
ResultsWindow	Класс, описывающий логику работы окна с результатами

Основные экранные формы

После запуска приложения на экране появляется главное окно приложения, изображённое на рисунке 3. Перед началом работы с приложением необходимо указать директорию со служебными файлами. Для этого необходимо нажать на кнопку «Выбрать» и выбрать в появившемся окне нужную директорию. После выбора директории в строке состояния, расположенной в нижней части главного окна, появляется надпись «Выполняется загрузка…».

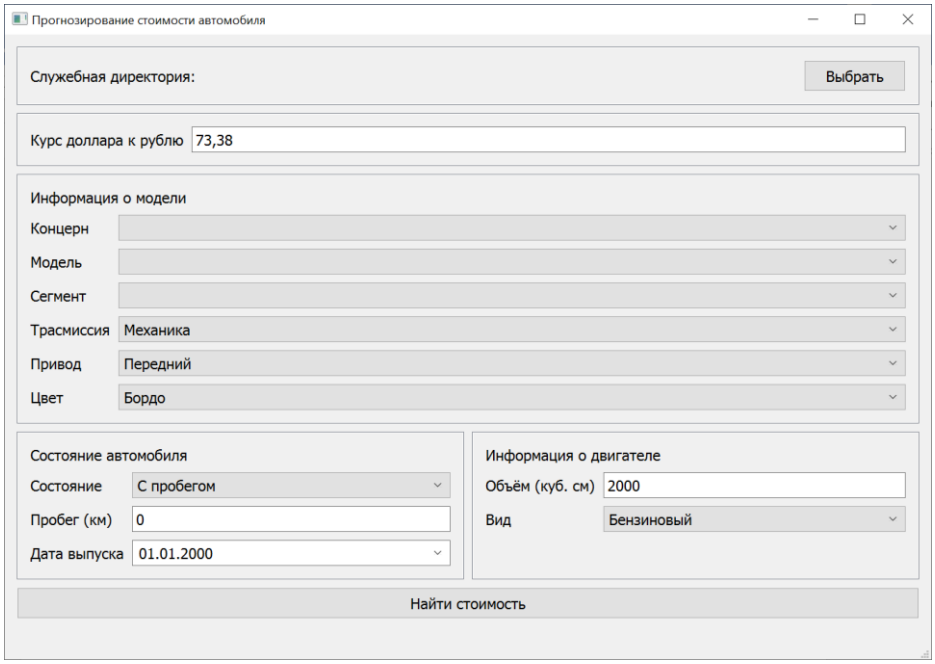

Рисунок 3 – Основное окно приложения

В случае, если один из служебных файлов не будет найден, пользователь получит сообщение об ошибке, изображённое на рисунке 4.

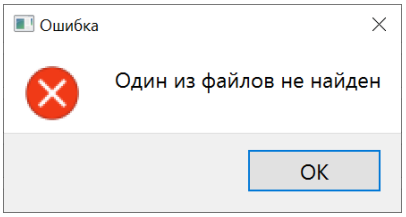

Рисунок 4 – Сообщение об ошибке

При попытке начать прогнозирование без указания служебной папки на экране появляется окно с ошибкой, представленное на рисунке 5.

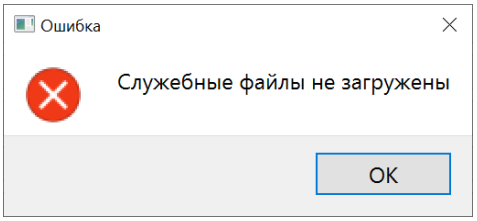

Рисунок 5 – Сообщение о необходимости загрузить служебные файлы

После загрузки служебных файлов и указания информации об автомобиле необходимо нажать на кнопку «Найти стоимость», после чего в строке состояния появляется надпись «Выполняется анализ…». По окончании анализа на экране появляется окно с результатами, изображённое на рисунке 6.

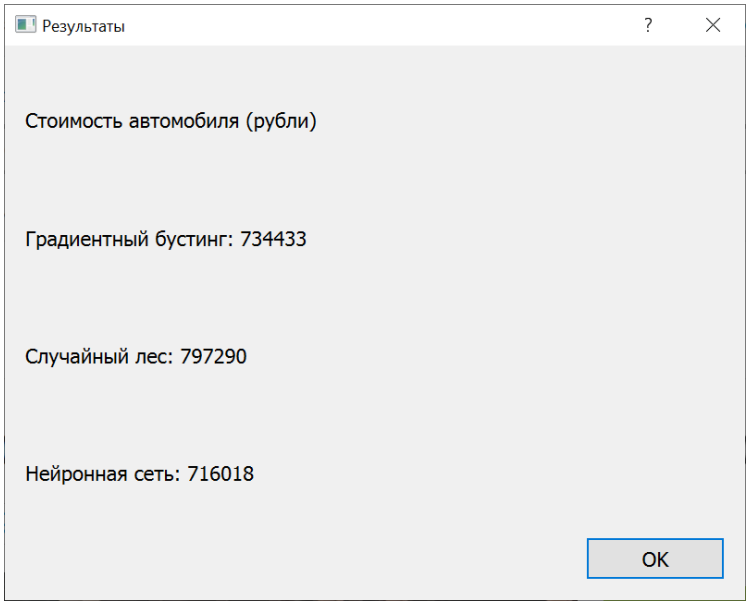

Рисунок 6 – Результаты прогнозирования

Анализ работы моделей

Для сравнения качества работы различных моделей были рассмотрены результаты работы приложения для ряда объявлений с интернет-сервиса Avito. В таблице 5 представлена информация о рассматриваемых объявлениях и предсказанные моделями стоимости.

Таблица 5 – Примеры объявлений и предсказанная стоимость

Характеристика	Пример №1	Пример №2	Пример №3
Производитель	Skoda	Renault	Mercedes-Benz
Модель	Rapid	19	E-класс
Год выпуска	2019	1997	2012
Состояние	С пробегом	Битый	С пробегом
Пробег, км	22000	200000	39800
Вид топлива	Бензин	Бензин	Бензин
Объём двигателя, $см^3$	1600	1400	1800
Цвет	Белый	Зелёный	Белый
Трансмиссия	Механика	Механика	Автомат
Тип привода	Передний	Передний	Задний
Сегмент	B	C	E
Стоимость в объявлении, рубли	845000	35000	1437000
Градиентный бустинг, рубли	734433	33088	1178316
Случайный лес, рубли	797290	42134	1078132
Нейронная сеть, рубли	716018	20540	1239772

При прогнозировании стоимости автомобиля из 1-го объявления наилучший результат показал случайный лес, при предсказании стоимости автомобиля из 2-го объявления наилучший прогноз выдал градиентный бустинг, а при предсказании стоимости автомобиля из 3-го объявления лучшей оказалась нейронная сеть.

Заключение

В ходе выполнения курсовой работы было реализовано программное обеспечение, позволяющее предсказывать стоимость подержанного автомобиля по его характеристикам.

Для решения задачи прогнозирования была применена многослойная нейронная сеть и 2 других подхода машинного обучения: градиентный бустинг над решающими деревьями и случайный лес. При проверке моделей на кросс-валидации наилучшее качество показала нейронная сеть. Однако в ходе анализа работы моделей на реальных данных было установлено, что в каждой конкретном случае лучшее качество показывают различные модели.

Обученные модели имеют хорошее качество работы. Несмотря на то, что иногда предсказания моделей отличаются от реальных данных, в этих случаях модели позволяют верно определить ценовую категорию, к которой относится автомобиль.

Список использованных источников

1 Belarus Used Cars Prices // Kagggle. – URL: https://www.kaggle.com/slavapasedko/belarus-used-cars-prices (дата обращения: 13.12.2020).

2 CatBoost // Викиконспекты университета ИТМО. – URL: https://neerc.ifmo.ru/wiki/index.php?title=CatBoost (дата обращения: 13.12.2020).

3 Introduction to Boosted Trees // Официальный сайт XGBoost. – URL: https://xgboost.readthedocs.io/en/latest/tutorials/model.html (дата обращения: 13.12.2020).

4 Random forest // Википедия. – URL: https://ru.wikipedia.org/wiki/Random_forest (дата обращения: 13.12.2020).

5 Случайный лес (Random Forest) // Анализ малых данных. – URL: https://dyakonov.org/2016/11/14/%D1%81%D0%BB%D1%83%D1%87%D0%B0%D0%B9%D0%BD%D1%8B%D0%B9-%D0%BB%D0%B5%D1%81-random-forest/ (дата обращения: 13.12.2020).

6 Neural network models (supervised) // Официальный сайт scikit-learn. – URL: https://scikit-learn.org/stable/modules/neural_networks_supervised.html (дата обращения: 13.12.2020).

7 Сайт объявлений Avito. – URL: https://www.avito.ru/ (дата обращения: 13.12.2020).