**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc165482677)

[1 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ 6](#_Toc165482678)

[1.1 Типы машинного обучения и способы обучения 6](#_Toc165482679)

[1.2 Линейная регрессия (Linear regression) 13](#_Toc165482680)

[1.3 Лассо регрессия (Lasso) 16](#_Toc165482681)

[1.4 Экстремальный градиентный бустинг (XGBoost) 18](#_Toc165482682)

[1.5 Случайный лес решений (Random Forest) 21](#_Toc165482683)

[1.6 Искусственная нейронная сеть 24](#_Toc165482684)

[2 РАЗРАБОТКА ПРОГРАММНОГО КОДА 36](#_Toc165482685)

[2.1 Сбор данных 36](#_Toc165482686)

[2.2 Первичная обработка данных 44](#_Toc165482687)

[2.3 Программная реализация методов машинного обучения 51](#_Toc165482688)

[2.4 Реализация графического интерфейса пользователя 64](#_Toc165482689)

[2.5 Тестирование разработанной программы 73](#_Toc165482690)

[ЗАКЛЮЧЕНИЕ 81](#_Toc165482691)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 82](#_Toc165482692)

[ПРИЛОЖЕНИЕ 85](#_Toc165482693)

[Приложение A. Листинг кода парсера 85](#_Toc165482694)

[Приложение Б. Листинг кода первичной обработки 89](#_Toc165482695)

[Приложение В. Листинг кода машинного обучения 91](#_Toc165482696)

[Приложение Г. Листинг кода графического интерфейса 94](#_Toc165482697)

**РЕФЕРАТ**

Разработка и реализация алгоритмов прогнозирования стоимости подержанных автомобилей с использованием методов машинного обучения, выпускная квалификационная работа: 100 с., 35 рисунков, 3 табл., 1 приложение, 25 источников.

Ключевые слова: МАШИННОЕ ОБУЧЕНИЕ, ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ, ИНТЕЛЛЕКТУАЛЬНЫЙ АНАЛИЗ ДАННЫХ, СЛУЧАЙНЫЙ ЛЕС.

Объект исследования – методы компьютерного моделирования и машинного обучения.

Предмет исследования – применение методов машинного обучения для прогнозирования цен на подержанные автомобили.

Цель работы – разработка приложения для прогнозирования стоимости подержанных автомобилей на основе методов машинного обучения.

Методами исследования являются: объектно-ориентированное проектирование, системный анализ, машинное обучение.

Полученные результаты: в ходе работы были проанализированы существующие методы машинного обучения, выбраны и реализованы наиболее подходящие для создания эффективной системы прогнозирования стоимости подержанных автомобилей. Было разработано программное обеспечение с графическим интерфейсом в виде Telegram-бота, упрощающее взаимодействие пользователя с системой.

Область применения: анализ и прогнозирование стоимости подержанных автомобилей. Разработанное приложение может быть адаптировано для применения в других областях, связанных с прогнозированием цен или других регрессионных задач.

# **ВВЕДЕНИЕ**

Машинное обучение – это раздел искусственного интеллекта, который изучает методы построения алгоритмов, способных обучаться на основе данных. Машинное обучение имеет множество приложений в различных областях, таких как биология, медицина, экономика, социология, психология и другие. Одной из важных задач машинного обучения является задача регрессии, которая заключается в нахождении зависимости между переменными на основе набора наблюдений. Решение задачи регрессии позволяет прогнозировать значения переменных, анализировать влияние факторов, определять закономерности и тенденции в данных.

Существует множество методов машинного обучения для решения задачи регрессии, которые отличаются по своей сложности, интерпретируемости, точности, скорости и требованиям к данным. Выбор оптимального метода зависит от конкретной постановки задачи, целей исследования и характеристик данных.

Актуальность данной работы обусловлена тем, что в последние годы в России наблюдается заметное увеличение спроса на подержанные автомобили. Это связано с рядом экономических и социальных факторов. Высокие цены, ограниченный ассортимент новых автомобилей, а также изменения в законодательстве сделали доступ к новым автомобилям более сложным. В результате потребители все чаще обращают внимание на рынок подержанных автомобилей, который предлагает более широкий выбор и более доступные цены.

Развитие рынка подержанных автомобилей в России происходит бурно и динамично. Увеличение количества сделок, рост интереса со стороны покупателей и продавцов, а также появление новых сервисов и платформ для торговли автомобилями с пробегом способствуют активизации этого сектора экономики. В таких условиях возникает потребность в разработке надежных инструментов для оценки стоимости подержанных автомобилей, что делает тему прогнозирования цен на подержанные автомобили с помощью методов машинного обучения особенно актуальной.

Цель: разработать приложение для прогноза стоимости подержанных автомобилей с помощью методов машинного обучения.

Задачи:

* Составить обзор существующих методов машинного обучения для задач регрессии, провести их сравнительный анализ, определить достоинства и недостатки
* Разработать программу для сбора данных с сайтов, предназначенных для размещения объявлений о продаже подержанных автомобилей.
* Провести первичную обработку собранных данных.
* Провести обучение выбранных моделей классификации и регрессии и оценить их точность.
* Разработать графический интерфейс для пользователей.
* Провести тестирование разработанного приложения.

1. **ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**
   1. **Типы машинного обучения и способы обучения**

Машинное обучение – это отрасль компьютерных наук, фокусирующаяся на создании алгоритмов, которые обучаются и адаптируются, используя данные для прогнозирования и принятия решений. В отличие от традиционных программ, которые следуют жёстко заданным инструкциям, алгоритмы машинного обучения настраиваются на основе обучающего набора данных, что позволяет им аппроксимировать сложные зависимости в данных [16].

Для настройки этих алгоритмов используются методы из разнообразных областей математики, включая статистику, оптимизацию, численный анализ, теорию вероятностей, линейную алгебру и другие. Итоговая модель представляет собой функцию, способную восстанавливать неявные закономерности в данных.

Машинное обучение сочетает в себе теоретические и практические аспекты. Оно требует экспериментальной проверки на реальных данных для оценки эффективности разрабатываемых методов, что делает его как математической, так и инженерной дисциплиной.

Есть три типа методов машинного обучения: дедуктивное, индуктивное и трансдуктивное [21].

Дедуктивный метод – это подход «от общего к частному», который использует заранее сформулированные экспертами знания. Эти знания переводятся в компьютерный формат, например, в виде баз данных знаний, уравнений или теорем. Экспертные системы, использующие дедуктивное обучение, способны генерировать новые факты и правила на основе этих предварительно заданных данных.

Индуктивный метод – это процесс «от частного к общему», где обучение происходит на основе анализа эмпирических данных, полученных через наблюдения или эксперименты. Этот подход тесно связан с машинным обучением и включает в себя выявление закономерностей для создания общих правил. Индуктивное обучение часто используется в качестве альтернативы традиционным статистическим методам и связано с извлечением информации и анализом данных.

Трансдуктивный метод, предложенный Владимиром Вапником, фокусируется на решении конкретных задач без формулирования общих закономерностей. Этот подход позволяет делать выводы о новых эмпирических данных, опираясь на уже имеющиеся данные, без необходимости обобщения или формализации знаний. Трансдукция особенно полезна, когда доступно недостаточно данных для создания общих моделей.

При решении задачи методом индукции, когда ищется общий ответ для всех возможных случаев, неразмеченные объекты не учитываются, их как бы нет для решающего задачу, потому что с точки зрения индуктивного обучения могут быть и другие неразмеченные объекты кроме имеющихся. Учёт присутствующих неразмеченных данных может кардинально изменить качество решения, но если появятся новые неразмеченные данные, то их появление может полностью изменить ответ. Трансдуктивное обучение применяется в некоторых методах машинного обучения с частичным привлечением учителя (semi-supervised learning). Взаимосвязь между тремя типами обучения можно увидеть на рисунке 1.1.

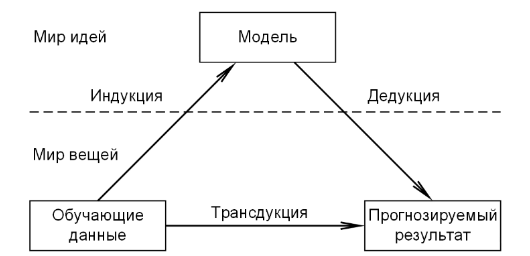


Рисунок 1.1 – Три типа обучения

Методы машинного обучения обычно разделяются на две обширные категории, в зависимости от наличия обучающего «сигнала» или «обратной связи» для алгоритма обучения: обучение с учителем (supervised learning) и обучение без учителя (unsupervised learning).

При обучении с учителем система обучается на примерах с заранее известными правильными ответами. На основе этих входных примеров и известных правильных ответов требуется восстановить зависимость между множеством примеров и множеством ответов, т.е. построить алгоритм, который будет выдавать достаточно точный ответ для любого примера. Совокупность примеров (входных объектов) и соответствующих им правильных ответов называется обучающей выборкой. Пусть обучающая выборка описывается парой значений 〈*x, y*〉, где – это данные (многомерный вектор признаков), y – это целевое значение (метка или правильный ответ). Надо найти функцию *ƒ(x) = y*.

Обучение без учителя, самообучение, происходит на примерах без заранее известных правильных ответов. Система сама находит внутренние взаимосвязи, зависимости, закономерности, существующие между объектами без вмешательства внешнего учителя, экспериментатора, человека. Пусть каждый объект описан вектором признаков . Надо найти механизм, который описывает структуру этих данных, которая заранее не известна.

Комбинированные виды обучения применяют различные сочетания обоих типов обучения в одной программе. Например, обучение с частичным привлечением учителя, обучение с подкреплением и некоторые другие.

При обучении с подкреплением учителем является сама окружающая среда, модель среды или неявный учитель, например, одновременная активность нескольких нейронов в искусственной нейронной сети.

Не всегда удаётся найти хорошую обучающую выборку. Часто данные размечены не полностью, т.е. не для всех данных есть правильный ответ (метка). Разметка данных для машинного обучения является однообразным и долгим трудом. Обычно имеется небольшое количество размеченных данных и большое количество неразмеченных данных. В этом случае применяется обучение с частичным привлечением учителя. Его ещё называют полуавтоматическим обучением (semi-supervised learning). Многие исследователи машинного обучения обнаружили, что неразмеченные данные, при использовании в сочетании с небольшим количеством размеченных данных, могут значительно улучшить точность обучения. Обучение с частичным привлечением учителя является частным случаем трансдуктивного обучения.

Методы машинного обучения разделяются по типам решаемых задач: классификация, кластеризация, регрессия, прогнозирование, идентификация, восстановление плотности распределения вероятности по набору данных, понижение размерности, одноклассовая классификация и выявление новизны, построение ранговых зависимостей и т.д. Продолжают возникать новые типы задач и даже целые новые дисциплины машинного обучения, например, добыча данных (data mining).

Классификация – разделение множества объектов или ситуаций на классы с помощью обучения с учителем. Классифицировать объект – значит, указать номер, имя или метку класса, к которому относится данный объект. Иногда требуется указать вероятность отношения объекта к классу. Например, по обучающей выборке фотографий котов и собак научиться различать изображения котов и собак.

Кластеризация (сегментация) – разделение множества объектов или ситуаций на кластеры с помощью обучения без учителя. Кластеризация (обучение без учителя) отличается от классификации (обучения с учителем) тем, что перечень групп четко не задан и определяется в процессе работы алгоритма, т.е. нет заранее определённых «правильных» ответов. Иногда указывается общее количество кластеров, но часто алгоритм сам выбирает оптимальное количество кластеров. Похожесть или близость объектов в кластере определяется через расстояние в многомерном пространстве признаков. Для этого нужно определить само пространство признаков (какие свойства измеряются) и метрику близости (как считается расстояние). Результаты кластеризации применяются при нахождении новых, ранее неизвестных знаний и зависимостей в данных (добыча данных или data mining). Например, задача нахождения целевой аудитории определённого товара путём анализа потребительских корзин покупателей с учётом пола, возраста, социального статуса, семейного положения и т.д.

Регрессия – нахождение зависимости выходной переменной от одной или нескольких независимых входных переменных с помощью обучения с учителем. В отличие он задач классификации, которые разделяют объекты на дискретное количество классов, задачи регрессии находят зависимости между непрерывными величинами. Например, нахождение зависимости между количеством съеденной пищи и весом тела.

Прогнозирование – это предсказание во времени. Прогнозирование похоже либо на регрессию, либо на классификацию в зависимости от данных задачи (непрерывные или дискретные данные), но в отличие от регрессии и классификации всегда направлено в будущее. В прогнозировании данные упорядочиваются по времени, которое является явным и ключевым параметром, а найденная зависимость экстраполируется в будущее. В прогнозировании применяются модели временных рядов.

Идентификация. Идентификация и классификация многими ошибочно понимаются как синонимы. Задача идентификации исторически возникла из задачи классификации, когда вместо определения класса объекта потребовалось уметь определять, обладает объект требуемым свойством или нет. Особенностью задачи идентификации является то, что все объекты принадлежат одному классу, и не существует возможности разделить класс на подклассы, т.е. сделать состоятельную выборку из класса, которая не будет обладать требуемым свойством. Если требуется определить человека по фотографии его лица, причём множество запомненных в базе людей постоянно меняется и появляются люди, которых не было в обучающем множестве, то это задача идентификации, которая не сводится к задаче классификации. В случае определения объекта по фотографии функция идентификации ) принимает в качестве аргументов два вектора признаков фотографий, а на выходе равна либо 1 в случае фотографий одного и того же объекта либо 0 в случае фотографий разных объектов одного и того же класса.

Восстановление плотности распределения вероятности по набору данных (kernel density estimate). Данная задача является центральной проблемой математической статистики. Математическая статистика решает обратные задачи: по результату эксперимента определяет свойства закона распределения. Исчерпывающей характеристикой закона распределения является плотность распределения вероятностей. Например, известен возраст людей, берущих кредит в банке, требуется найти плотность распределения вероятности возрастов заёмщиков.

Понижение размерности данных и их визуализация. Является частным случаем кластеризации. Каждый объект может быть представлен в виде многомерного вектора признаков , нужно получить более компактное признаковое описание объекта ., где *k < n*. Понижение размерности может помочь другим методам путём устранения избыточных данных. Используется при разведочном анализе и для устранения «проклятия размерности», когда данные быстро становятся разреженными при увеличении размерности пространства признаков. Например, дан список документов на естественном языке, требуется найти документы с похожими темами.

Одноклассовая классификация и выявление новизны. Или задача поиска аномалий, выбросов, которые не относятся ни к одному кластеру. Нахождение объектов, которые отличаются по своим свойствам от объектов обучающей выборки. Является задачей обучения без учителя. Например, обнаружение инородных предметов (кости, камни, кусочки упаковки) в продуктах питания при их сканировании рентгеновским сканером при неразрушающем контроле качества продукции, обнаружение подозрительных банковских операций, обнаружение хакерской атаки, медицинская диагностика и т.д.

Построение ранговых зависимостей. Ранжирование – это процедура упорядочения объектов по степени выраженности какого-либо качества в порядке убывания этого качества. Задачами ранжирования являются: сортировка веб-страниц согласно заданному поисковому запросу, персонализация новостной ленты, рекомендации товаров (видео, музыки), адресная реклама.

Добыча данных (data mining) или интеллектуальный анализ данных – совокупность методов обнаружения в данных ранее неизвестных, нетривиальных, практически полезных и доступных интерпретации знаний, необходимых для принятия решений в различных сферах человеческой деятельности. В данный момент добыча данных отделяется от машинного обучения в отдельную дисциплину.

Интеллектуальный анализ данных и машинное обучение имеют различные цели: машинное обучение прогнозирует на основе известных свойств, полученных от обучающей выборки, а интеллектуальный анализ данных фокусируется на добыче новых ранее неизвестных зависимостей в данных. Однако обе дисциплины используют одинаковые методы.

Различные типы задач схематически представлены на рисунке 1.2.

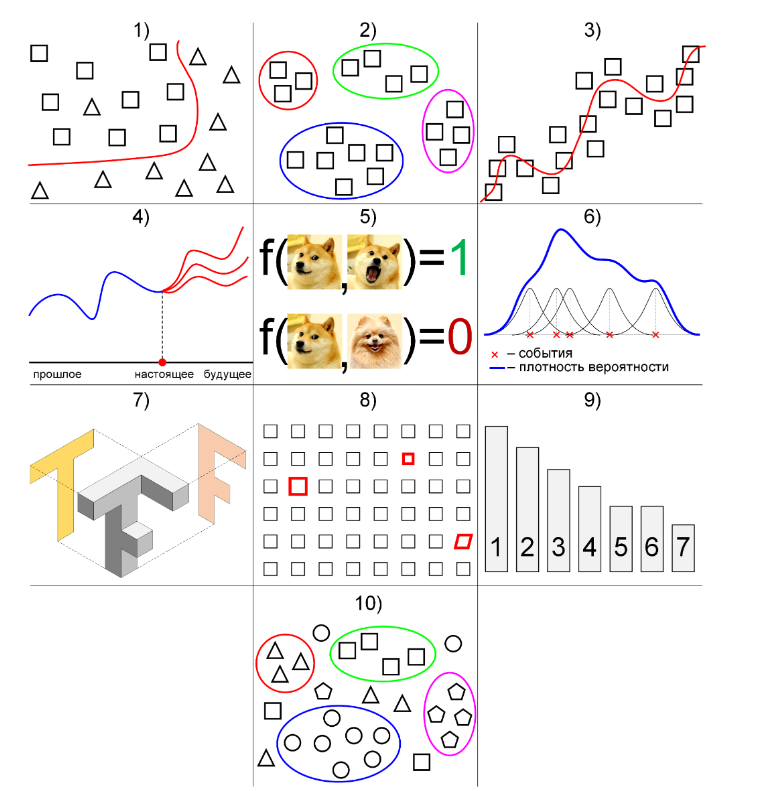


Рисунок 1.2 - Схематическое представление типов задач:  
1) классификация; 2) кластеризация; 3) регрессия; 4) прогнозирование; 5) идентификация;  
6) восстановление плотности распределения вероятности по набору данных;  
7) понижение размерности; 8) одноклассовая классификация и выявление новизны;  
9) построение ранговых зависимостей; 10) добыча данных.

В данной работе будет использоваться два типа задач: задача регрессии - прогноз на основе выборки объектов с различными признаками. На выходе должно получиться вещественное число (50, 100000, 7645.454 и др.) – цена автомобиля; Задача классификации – определение, к какому классу или категории относится объект. На выходе получается категориальный (номинальный) ответ, который указывает класс объекта.

* 1. **Линейная регрессия (Linear regression)**

Термин «линейность» в контексте алгебры означает прямую зависимость между двумя или более переменными. Когда мы изображаем это отношение в двухмерном пространстве, используя две переменные, результатом является линия.

Линейная регрессия — это метод прогнозирования значений одной переменной на основе другой. Этот метод выявляет прямую связь между входными данными (*x*) и результатом (*y*), отсюда и название «линейная» регрессия. При построении графика с независимой переменной по оси *x* и зависимой переменной по оси *y*, линейная регрессия создает линию, которая наилучшим образом соответствует набору данных, рисунок 1.3.

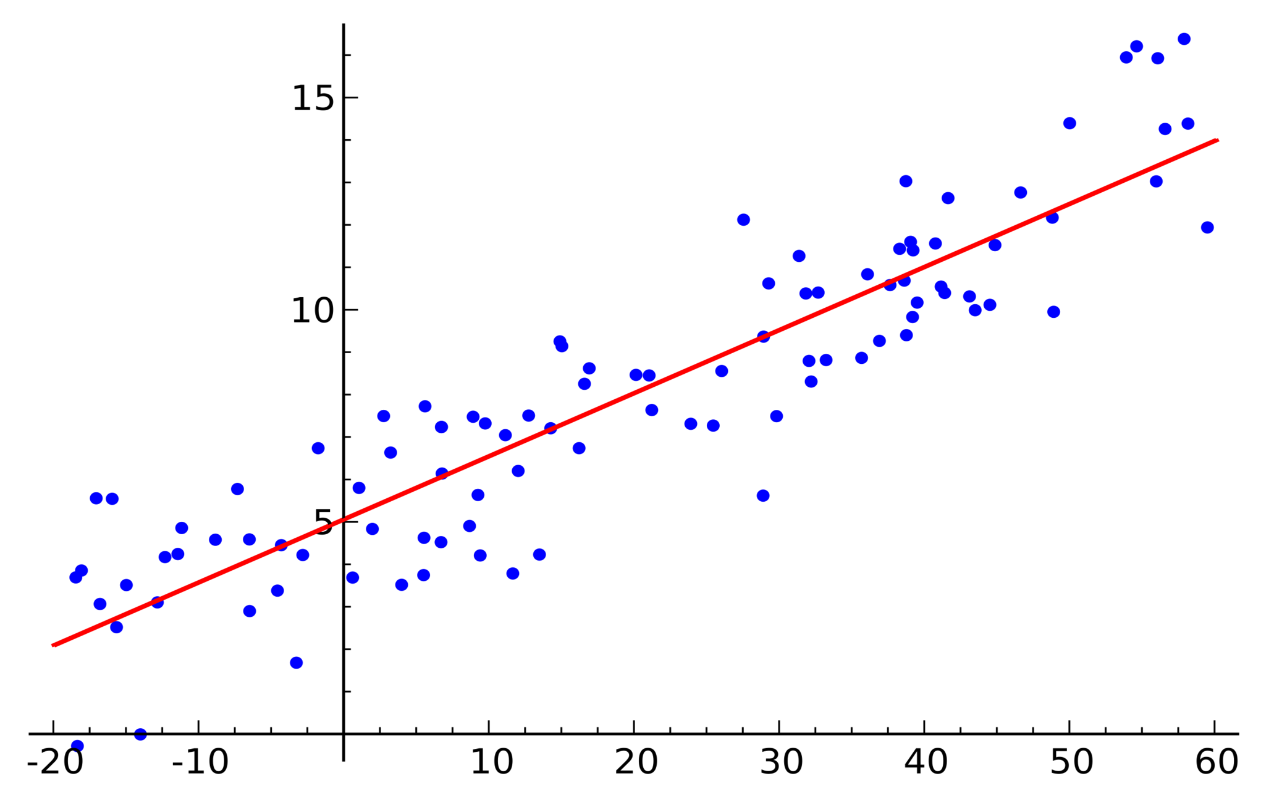


Рисунок 1.3 – Пример линейной регрессии

Уравнение линии имеет вид:

,

где *b* – точка пересечения с осью *y*, а *m* – угловой коэффициент прямой.

Линейная регрессия помогает найти оптимальные значения для этих параметров. Переменные y и x фиксированы, так как они представляют собой данные, в то время как точка пересечения и угловой коэффициент могут быть подобраны так, чтобы минимизировать ошибку прогноза.

Эта концепция расширяется до множественной линейной регрессии, когда необходимо учесть больше двух переменных. Например, если нужно предсказать стоимость автомобиля на основе его мощности, цвета, года выпуска и марки автомобиля, то зависимая переменная будет функцией нескольких независимых переменных. В таких случаях модель представляет собой гиперплоскость, которая в двумерном пространстве является линией, в трехмерном – плоскостью, а в пространстве с большим количеством измерений – гиперплоскостью [22].

Модель регрессии с участием нескольких переменных может быть представлена как:

,

где *y* – целевая переменная, значение которой требуется предсказать; – свободный член, который представляет собой значение y, когда все независимые переменные равны нулю; – независимые переменные, которые используются для предсказания целевой переменной; – коэффициенты при соответствующих независимых переменных, которые показывают насколько сильно каждая независимая переменная влияет на целевую переменную. Эти коэффициент определяются в процессе обучения.

Алгоритм обучения линейной регрессии включает следующие шаги:

Шаг 1. Определение функции потерь.

Функция потерь *L* (также называемая функцией стоимости или ошибкой) измеряет, насколько хорошо модель предсказывает целевую переменную *y*. О том какие существуют функции потерь и какие будут использоваться в этой работе, будет описано позже.

Шаг 2. Вычисление прогнозируемых значений.

Прогнозируемые значения вычисляются как линейная комбинация признаков и параметров модели.

Шаг 3. Проверка минимизации функции потерь.

Проверяется, достигла ли функция потерь минимума. Если нет, переходят к следующему шагу.

Шаг 4. Обновление параметров модели

Параметры модели (веса) обновляются итеративно с использованием метода градиентного спуска. Веса обновляются в направлении антиградиента функции потерь:

где – коэффициент обучения; – частная производная функции потерь по.

Шаг 5. Повторение шагов 2–4

Повторяем шаги 2– 4 до тех пор, пока функция потерь не достигнет своего минимума или не перестанет изменяться существенно.

Достоинства линейной регрессии:

* Простота и интерпретируемость: легко реализовать и интерпретировать результаты.
* Быстрота обучения: обучение линейной регрессии относительно быстрое.
* Малое количество гиперпараметров: меньше параметров для настройки, чем у более сложных моделей.
* Работа с высокоразмерными данными: хорошо справляется с данными, где количество признаков больше количества наблюдений.

Недостатки линейной регрессии:

* Линейные зависимости: предполагает линейную зависимость между независимыми и зависимой переменными.
* Чувствительность к выбросам: выбросы могут сильно влиять на модель.
* Мультколлинеарность: высокая корреляция между независимыми переменными может вызвать нестабильность модели.
* Необходимость предварительной обработки данных: требуется масштабирование и нормализация данных, а также обработка категориальных переменных.
  1. **Лассо регрессия (Lasso)**

Метод наименьших квадратов (МНК) иногда может быть подвержен нестабильности, особенно когда модель слишком точно подгоняется под обучающие данные, что приводит к переобучению. Чтобы противостоять этому, используется регуляризация, которая вводит дополнительные ограничения на параметры модели, чтобы избежать её чрезмерной сложности. Это достигается путём “сжатия” коэффициентов β в процессе оптимизации, делая их в среднем меньше по абсолютной величине, чем при использовании МНК [24].

Лассо-регрессия (LASSO, Least Absolute Shrinkage and Selection Operator) вводит элемент регуляризации в функцию потерь, что помогает стабилизировать решение. Минимизация ошибки модели включает в себя регуляризационный параметр λ, который служит штрафом за сложность модели. Формула для оптимизации выглядит следующим образом:

Это позволяет найти баланс между точностью модели и количеством признаков, используемых для предсказания, сокращая некоторые коэффициенты до нуля и тем самым выбирая наиболее значимые признаки. При λ = 0, лассо-регрессия сводится к МНК, но по мере увеличения λ, модель упрощается до тех пор, пока не станет совсем простой, или даже нулевой. Оптимальное значение λ находится через перекрестную проверку, где оно соответствует наименьшей ошибке на данных, не использованных при обучении.

Регуляризация – это ключевой элемент в построении робастных моделей машинного обучения. Она не только предотвращает переобучение, но и способствует созданию более простых и понятных моделей, что особенно важно в условиях работы с большими и сложными наборами данных. Это делает регуляризацию ценным инструментом в арсенале специалиста по анализу данных.

Алгоритм обучения Лассо регрессии включает следующие шаги:

Шаг 1. Определение функции потерь.

Шаг 2. Вычисление прогнозируемых значений.

где – свободный член; – веса признаков.

Шаг 3. Проверка минимизации функции потерь

Проверяется, достигла ли функция потерь минимума. Если нет, переходят к следующему шагу.

Шаг 4. Обновление параметров модели

Параметры модели обновляются итеративно с использованием метода градиентного спуска. Веса обновляются в направлении антиградиента функции потерь. Формула такая же как и для модели линейной регрессии.

Шаг 5. Повторение шагов 2–4

Повторяем шаги 2–4 до тех пор, пока функция потерь не достигнет своего минимума или не перестанет изменяться существенно.

Достоинства лассо регрессии:

* Регуляризация и отбор признаков: лассо регрессия может вынудить некоторые коэффициенты стать равными нулю, эффективно выполняя отбор признаков и уменьшение размерности.
* Снижение переобучения: регуляризация помогает уменьшить риск переобучения, особенно при большом количестве признаков.
* Интерпретируемость: модель становится более интерпретируемой, так как некоторые коэффициенты могут быть равны нулю.

Недостатки лассо регрессии:

* Сложность вычислений: лассо регрессия требует большего количества вычислений по сравнению с обычной линейной регрессией, особенно для больших наборов данных.
* Чувствительность к выбору λ: производительность модели сильно зависит от правильного выбора коэффициента регуляризации λ. Слишком большое значение λ может привести к недообучению, а слишком маленькое – к переобучению.
* Работа с сильно коррелированными признаками: если признаки сильно коррелированы, Лассо регрессия может случайно выбрать один из них и установить другие в ноль, что может не всегда быть желаемым.
  1. **Экстремальный градиентный бустинг (XGBoost)**

Экстремальный градиентный бустинг или XGBoost, представляет собой передовой алгоритм машинного обучения, который используется для решения задач регрессии и классификации. Он основан на принципе градиентного бустинга, который заключается в последовательном улучшении модели путём добавления новых моделей, каждая из которых исправляет ошибки предыдущих. В случае XGBoost эти новые модели представляют собой деревья решений.

В отличие от традиционного градиентного бустинга, XGBoost внедряет ряд оптимизаций, которые ускоряют обучение и повышают эффективность, такие как:

* Регуляризация: XGBoost включает в себя L1 (Лассо) и L2 (Ридж) регуляризацию, что помогает предотвратить переобучение и делает модель более устойчивой.
* Обработка пропущенных значений: Алгоритм способен обрабатывать пропущенные данные, автоматически находя наилучший способ их обработки.
* Масштабируемость: XGBoost оптимизирован для работы на больших наборах данных и распределённых системах.
* Подрезка деревьев: Алгоритм использует стратегию подрезки деревьев, которая удаляет части дерева, не приносящие значительного улучшения модели.

Основное уравнение для XGBoost можно представить следующим образом:

где – предсказанное значение для i-го наблюдения, – *k*-я функция в ансамбле, – вектор признаков *i*-го наблюдения, *K* – количество деревьев в модели, и – пространство всех возможных моделей (в данном случае деревьев).

XGBoost широко используется в индустрии и научных исследованиях благодаря своей высокой производительности и способности обрабатывать различные типы данных. Он особенно ценится за точность предсказаний и эффективность на больших объёмах данных. Этот алгоритм стал золотым стандартом в соревнованиях по машинному обучению и часто используется как основа для создания сложных предсказательных моделей.

Алгоритм работы экстремального градиентного бустинга:

Шаг 1. Инициализация модели.

Инициализируем модель, используя начальное предсказание. Обычно это просто среднее значение целевой переменной для задачи регрессии или логарифмические вероятности для задачи классификации.

Шаг 2. Вычисление остаточной ошибки.

Для каждого наблюдения вычисляем остаточную ошибку, которая представляет собой разницу между реальными значениями и текущими предсказаниями модели.

Шаг 3. Построение нового дерева.

Строим новое дерево, которое будет предсказывать остаточную ошибку. Деревья строятся последовательно, каждое новое дерево старается уменьшить ошибку предыдущих деревьев.

Шаг 4. Обновление предсказаний.

Обновляем предсказания модели, добавляя предсказания нового дерева, умноженные на коэффициент обучения:

где – текущее предсказание для наблюдения *i* на итерации t; – предсказание на предыдущей итерации; – коэффициент обучения; – предсказание нового дерева для наблюдения *i.*

Шаг 5. Проверка остановки.

Проверяем условия остановки (например, количество деревьев или улучшение функции потерь). Если условия не выполнены, возвращаемся к шагу 2.

Шаг 6. Итерация процесса.

Процесс продолжается до тех пор, пока не будет достигнуто максимальное количество итераций или улучшение функции потерь станет незначительным.

Достоинства XGBoost:

Высокая точность: XGBoost часто показывает высокую точность на различных задачах и является фаворитом на соревнованиях по машинному обучению.

Эффективность вычислений: XGBoost оптимизирован для высокой производительности и может обрабатывать большие наборы данных благодаря методам параллельной и распределенной обработки.

Регуляризация: включает L1 и L2 регуляризацию, что помогает избежать переобучения и улучшить обобщающую способность модели.

Гибкость: поддерживает множество функций настройки, таких как различное количество деревьев, их глубина, скорость обучения и множество других гиперпараметров.

Обработка пропущенных значений: XGBoost умеет автоматически обрабатывать пропущенные значения, что упрощает предварительную обработку данных.

Важность признаков: XGBoost может оценивать важность признаков, что помогает в интерпретации модели.

Недостатки XGBoost:

Сложность настройки: большое количество гиперпараметров требует тщательной настройки, что может быть трудоемким процессом.

Ресурсоемкость: XGBoost может потреблять много вычислительных ресурсов, особенно при работе с большими наборами данных и сложными моделями.

Чувствительность к выбросам: хотя XGBoost менее чувствителен к выбросам, чем некоторые другие модели, он все же может быть подвержен их влиянию.

Время обучения: время обучения может быть значительным, особенно при работе с большими наборами данных и без соответствующей оптимизации гиперпараметров.

* 1. **Случайный лес решений (Random Forest)**

Случайный лес — это ансамблевый метод машинного обучения, который представляет собой усовершенствованную форму деревьев решений. Этот метод разработан для устранения основных недостатков деревьев решений, особенно их склонности к переобучению. Случайный лес объединяет множество деревьев решений для создания более мощной и надежной модели, которая обеспечивает высокую точность предсказаний и лучшую обобщающую способность.

Случайный лес строит множество деревьев решений, каждое из которых обучается на случайно выборке данных из общего обучающего набора. Эти деревья используют различные наборы признаков, что обеспечивает высокое разнообразие внутри ансамбля. В задачах классификации итоговый ответ модели формируется путем голосования между всеми деревьями, где выбирается наиболее частый результат. Для регрессии же итоговое значение предсказания получается путем усреднения ответов от всех деревьев [20]. Принцип работы случайного леса иллюстрируется на рисунке 1.4.

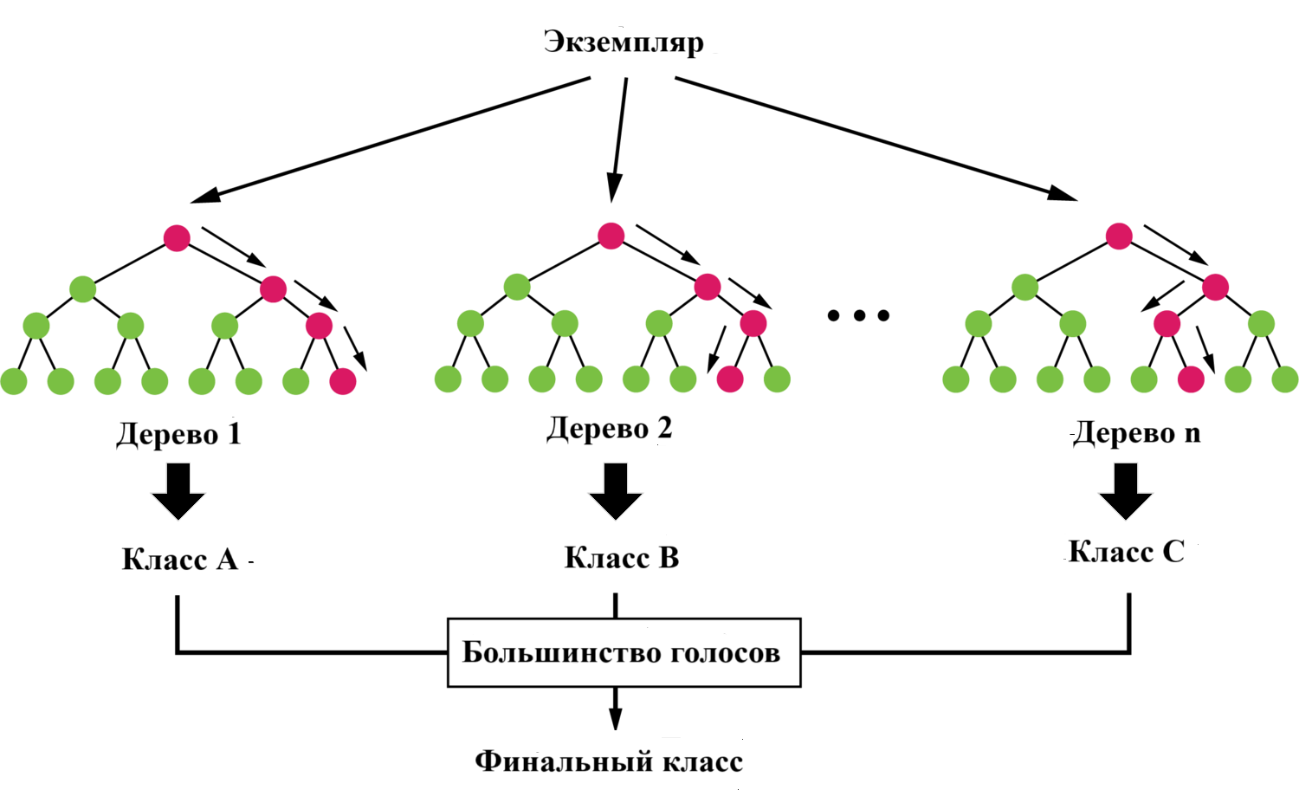


Рисунок 1.4 – Принцип работы случайного леса

Основные принципы работы случайного леса:

* Ансамбль деревьев: случайный лес создаёт множество деревьев решений, каждое из которых обучается на случайной выборке данных с использованием случайного подмножества признаков.
* Уменьшение переобучения: благодаря стратегии «мудрости толпы», случайный лес снижает риск переобучения, который часто встречается в одиночных деревьях решений.
* Важность признаков: алгоритм может оценить, какие признаки наиболее важны для прогнозирования, что делает его полезным для понимания данных.
* Гибкость: случайный лес может быть использован как для задач классификации, так и для регрессии.

Математически алгоритм случайного леса можно описать следующим образом:

1. Для *b* = 1, 2, …, *B*, где *B* – количество деревьев:
2. Извлечь выборку *S* размера *N* из обучающего набора данных
3. По выборке *S* построить полное дерево , рекурсивно повторяя следующие шаги для каждого терминального узла, пока не будет достигнуто минимальное количество наблюдений в нём
4. Из первоначального набора *M* предикторов случайно выбрать m предикторов
5. Из *m* предикторов выбрать предиктор, который обеспечивает наилучшее расщепление
6. Расщепить узел на два узла потомка

2. В результате получится ансамбль деревьев решений

3. Пусть – класс, спрогнозированный деревом решений , то есть ; тогда – это класс, наиболее часто встречающийся в множестве

Преимущества случайного леса:

* Высокая точность: Сочетание множества деревьев решений обычно приводит к более точным и надёжным предсказаниям по сравнению с одиночными деревьями.
* Устойчивость к шуму: Модель устойчива к шуму в данных и способна обрабатывать большие наборы данных с большим количеством признаков.
* Обработка несбалансированных данных: Случайный лес эффективно работает с несбалансированными наборами данных, что делает его подходящим для широкого спектра приложений.

К недостаткам Случайного леса можно отнести:

* Объёмные модели: Ансамбли, состоящие из множества деревьев, могут занимать значительное количество памяти.
* Затраты времени на обучение: Создание и настройка большого количества деревьев может потребовать длительного времени, особенно при работе с большими объёмами данных.
* Сложность интерпретации: По мере увеличения количества деревьев в лесу, понимание того, как модель приходит к конкретному решению, становится менее очевидным.
  1. **Искусственная нейронная сеть**

Искусственная нейронная сеть (ИНС), по сути, является математической моделью сети нейронов человеческого мозга. Это позволяет нам думать о нейронах ИНС как об активированных или неактивированных, а также об их взаимодействии, подобном эффекту бабочки, когда включение в работу каждого отдельного нейрона зависит от состояний всех его сетевых предшественников [1]. Строение искусственного нейрона показано на рисунке 1.5.

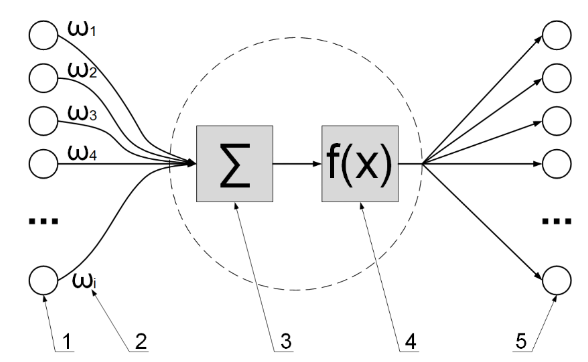


Рисунок 1.5 – Схема искусственного нейрона

1) нейроны, выходные сигналы которых поступают на вход данному;  
2)  – веса входных сигналов; 3) сумматор входных сигналов;  
4) вычислитель передаточной (активационной) функции;  
5) нейроны, на входы которых подаётся выходной сигнал

На вход искусственного нейрона поступают импульсы от произвольного числа других нейронов сети. Связи, по которым выходные сигналы одних нейронов поступают на входы других, часто называют синапсами по аналогии со связями между биологическими нейронами. Каждая связь характеризуется своим весом. Связи с положительным весом называются возбуждающими, а с отрицательным – тормозящими. Нейрон имеет один выход, часто называемый аксоном по аналогии с биологическим прототипом. С единственного выхода нейрона сигнал может поступать на произвольное число входов других нейронов. Если на выходе нейрона есть ненулевой сигнал (положительный или отрицательный), то говорят, что нейрон активен или возбуждён.

В сумматоре поступающие импульсы на вход нейрона умножаются на веса входов и слаживаются:

где *n* – количество входящих синапсов;  – веса входов (положительные возбуждающие или отрицательные тормозящие);  – сигналы на входах; C – константа для формирования порога чувствительности нейрона, которая называется сдвиг (bias). Функция x называется индуцированным локальным полем нейрона или взвешенной суммой. Возможные значения сигналов на входах нейрона  считают заданными в интервале [0,1]. Сигналы на входах, в зависимости от архитектуры сети, могут быть либо дискретными (только 0 или только 1), либо аналоговыми (непрерывными в интервале от 0 до 1 включительно).

Затем к индуцированному локальному полю нейрона x применяется функция, называемая передаточной функцией *ƒ(x)* или функцией активации, функцией срабатывания, которая определяет зависимость сигнала на выходе нейрона от взвешенной суммы сигналов на его входах. Использование различных передаточных функций позволяет вносить нелинейность в работу нейрона и в целом нейронной сети. Без этой нелинейности нейронная сеть вырождается в задачу линейной алгебры, т.е. в обычное перемножение матриц и векторов.

Существует множество различных передаточных функций. В таблице 1.1 показаны несколько самых широко применяемых в ИНС функций активации.

Таблица 1.1 – Функции активации нейрона в ИНС

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название | Формула | Изображение |
| 1 | Rectified linear unit (ReLU) |  |  |
| 2 | Параметрический rectified linear unit (PReLU) |  |  |
| 3 | Логистическая функция или сигмоида |  |  |
| 4 | Гиперболический тангенс (TanH) |  |  |
| 5 | Арктангенс (ArcTan) |  |  |

Здесь аргумент активационной функции x – это взвешенная сумма или индуцированное локальное поле; α во второй строке таблицы 1.1 – это малый коэффициент, обычно равный α = 0,01. Малый коэффициент *α* нужен, чтобы производная функции ReLU не была равна нулю при *x* < 0. Однако наличие малого коэффициента α некритично, как может показаться, так как вероятность того, что активационные функции всех нейронов ИНС станут равными нулю крайне мала. Например, в одномерном случае невозможно преодолеть гору, в двумерном случае гору можно обойти слева или справа, а в многомерном *N*-мерном пространстве существует множество путей обхода препятствий, и чем больше измерений, тем меньше вероятность попасть в локальный минимум.

Функция ReLU (строка 1 таблицы 2.1) была впервые применена в свёрточных ИНС для устранения проблемы исчезающего градиента и доказала свою эффективность. В отличие от сигмоиды и гиперболического тангенса, операция ReLU гораздо быстрее, при этом качество обучения снижается незначительно. Поэтому ReLU на данный момент является самой популярной активационной функцией в свёрточных ИНС [12].

Функция активации необходима для внесения нелинейности в систему. Без нелинейности ИНС будет вырождаться в обычное перемножение матриц и векторов, т.е. в линейную алгебру. В данном контексте уместно напомнить общеизвестный факт о том, что никакие линейные преобразования пространства признаков не позволяют улучшить качество классификации, т.е. линейная разделимость объектов в пространстве признаков является неэффективной. С помощью линейных преобразований невозможно решить нетривиальные задачи, например, разделить пересекающиеся множества.

Функция потерь в нейронных сетях играет ключевую роль. Она используется для оценки разницы между реальными и предсказанными ответами. Главная цель — минимизировать эту ошибку, чтобы обучение нейронной сети приближалось к желаемой цели. Функция потерь измеряет, насколько хорошо нейронная сеть работает на обучающей выборке по сравнению с ожидаемыми ответами. Она может зависеть от различных параметров, таких как веса и смещения [4].

Существует несколько известных функций потерь:

* Средняя квадратичная ошибка (MSE): широко используется в линейной регрессии. Для ее расчета берется разница между предсказанными и истинными значениями, возводится в квадрат и усредняется по всему набору данных [7].
* Кросс-энтропия: часто используется в задачах классификации.
* Функция потерь для бинарной классификации.
* Коэффициент детерминации R2: показывает долю дисперсии зависимой переменной, объясненную независимыми переменными в модели линейной регрессии.
* Средняя абсолютная ошибка (MAE): измеряет среднее значение абсолютных разностей между предсказанными и истинными значениями.

В данной работе для задачи регрессии используются:

Средняя квадратичная ошибка (MSE):

где – предсказанное значение моделью для *i*-го наблюдения; – фактическое значение целевой переменной для *i*-го наблюдения; *m* – количество наблюдений в наборе данных.

Коэффициент детерминации R2:

где – фактическое значение целевой переменной для *i*-го наблюдения; – предсказанное значение моделью для i-го наблюдения; – среднее значение y по всем наблюдениям; *m* – количество наблюдений.

Средняя абсолютная ошибка (MАE):

где – абсолютная разница между предсказанным и истинным значением для каждого наблюдения; *m* – общее количество наблюдений.

Для задачи классификации в данной работе используется разреженная категориальная перекрестная энтропия (SCCE) [17]:

,

где – softmax функция или нормализованная экспонента; *m* – общее количество наблюдений.

Обучение нейросетей представляет собой процесс, который можно разделить на два основных этапа:

* прямое распространение ошибки
* обратное распространение ошибки

Рассмотрим каждый из этих этапов более подробно.

Прямое распространение ошибки (Forward Propagation), представлено на рисунке 1.6.

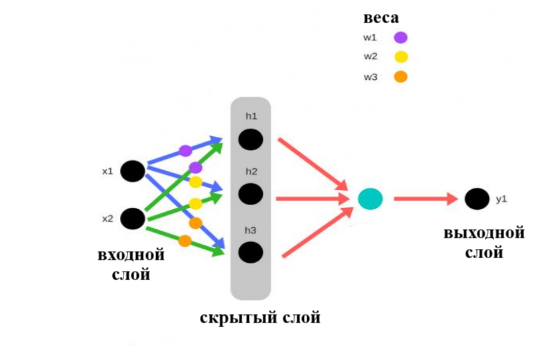


Рисунок 1.6 – Прямое распространение ошибки

На этом этапе происходит вычисление предсказания нейросетью. Входные данные подаются на вход слоев нейросети, где каждый следующий слой преобразует их с использованием определённых весов и биасов, которые были инициализированы в начале обучения или переданы от предыдущих итераций обучения. В каждом слое применяются активационные функции, которые добавляют нелинейность в обработку данных, позволяя тем самым нейросети учиться и моделировать более сложные зависимости между входными и выходными данными.

Процесс прямого распространения продолжается до тех пор, пока входные данные не пройдут через все слои сети и в результате будет сформирован выходной сигнал. Этот выходной сигнал является предсказанием нейросети, которое затем сравнивается с фактическими значениями для вычисления ошибки предсказания.

Обратное распространение ошибки (Backpropagation), представлено на рисунке 1.7.

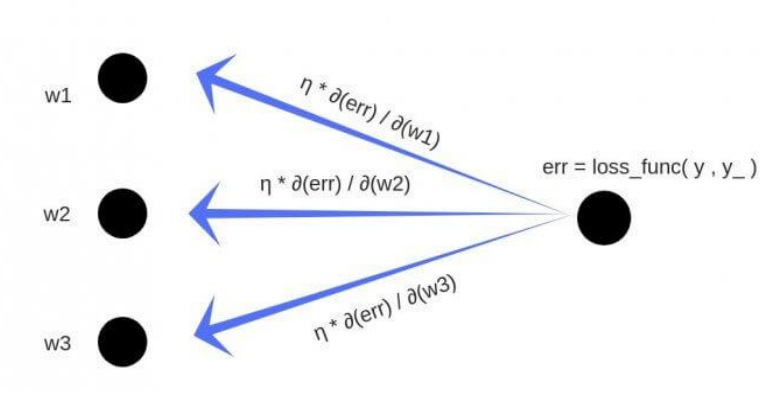


Рисунок 1.7 – Обратное распространение ошибки

Метод обратного распространения ошибки (backpropagation) – это метод вычисления градиента, который используется при автоматическом обновлении весов многослойного перцептрона. Метод является частным случаем более общей техники автоматического дифференцирования (automatic differentiation) и модификацией классического метода градиентного спуска (gradient descent) [14].

Для метода обратного распространения ошибки прежде всего требуется построить функцию потерь или функцию оценки работы сети H. Простейшим и самым распространённым примером функции потерь является квадратичное отклонение:

где *H* – функция потерь;  – сигнал на *i*-том выходном нейроне ИНС;  – ожидаемый сигнал при правильном ответе на i-том выходном нейроне ИНС. Сумма берётся по всем выходным нейронам сети от 1 до n. Таким образом при квадратичном отклонении метод обратного распространения ошибки сводится к методу наименьших квадратов. То есть веса синапсов  и порог активации каждого нейрона *, k* в ИНС должны быть такими, чтобы сумма была минимальной, желательно, нулевой или близкой к нулю. Здесь *j* – номер нейронного слоя, *k* – номер нейрона в слое, *l* – номер связи в нейроне.

Метод наименьших квадратов не всегда является наилучшим выбором функции потерь. Умение конструировать оптимальную функцию потерь может существенно улучшить качество распознавания и на порядок ускорить обучение ИНС.

Теоретически можно подобрать оптимальные веса синапсов и порогов активации нейронов с помощью простого перебора, через прямое вычисление без применения эффективных приёмов (brute force computation). Однако на практике для многослойных глубоких сетей время такого прямого вычисления будет намного порядков превышать время жизни Вселенной.

Каждый синапс и каждый порог нейрона можно считать отдельным измерением некоего многомерного пространства. В глубокой ИНС таких синапсов и порогов тысячи, а иногда и десятки миллионов. При увеличении количества синапсов пространство становится многомерным и появляется проблема под названием «проклятие размерности», которая связанна с экспоненциальным возрастанием данных из-за увеличения размерности пространства. Например, чтобы покрыть единичный отрезок [0, 1] точками с частотой 0,01, требуется 100 точек. Чтобы покрыть квадрат с такой же частотой, требуется 104 точек. Для десятимерного куба требуется уже 102 \* 10 = 1020 точек, что в 1018 раз больше, чем в одномерном пространстве. Таким образом использование переборных (brute force) алгоритмов становится неэффективным при возрастании размерности системы.

Алгоритм обратного распространения ошибки состоит из пяти шагов.

Шаг 1. Конструируется функция потерь *H*, которая совсем не обязательно должна быть квадратичным отклонением

Шаг 2. Высчитывается величина ошибки

Шаг 3. Проверяется, достигла ли функция потерь минимума

Шаг 4. Если функция потерь не достигла минимума, тогда параметры ИНС, т.е. веса синапсов и пороги активации, обновляются по итеративной формуле:

*=*

*=*

где  – вес синапса;   – порог активации нейрона;  *H* – функция потерь;  0 < *η* < 1 – множитель, задающий скорость обучения;  *j* – номер нейронного слоя; *k* – номер нейрона в слое;  *l* – номер связи в нейроне. Для обновления параметров ИНС нужно двигаться в обратном направлении от выходов ко входам сети и вычислять градиенты от функции потерь для каждого веса и порога. Обновлённые веса и пороги участвуют в обновлении следующих в обратном порядке слоёв ИНС.

Шаг 5. Повторять шаги 2–4 пока функция потерь не достигнет своего минимума. Когда функция потерь достигла минимума, ИНС считается обученной и готова к работе с новыми данными.

Величина шага коррекции ошибки задаётся с помощью множителя 0 < η < 1. Шаг коррекции ошибки не должен быть слишком маленьким, иначе обучение будет долгим. Шаг коррекции ошибки не должен быть слишком большим, иначе можно «перешагнуть» через минимум, из-за чего в сети может возникнуть паралич (слишком большие веса) или неустойчивость (невозможность найти минимум функции потерь). Обычно множитель η не является константой и выбирается автоматически по следующему принципу: если градиент большой – множитель η маленький; если градиент маленький – множитель η большой.

При вычислении градиентов на шаге 4 алгоритма применяется цепное правило дифференцирования сложной функции (chain rule):

где *ƒ* и *g* – функции переменной *x*.

Также применяется уравнение частной производной:

где ƒ – функция трёх переменных *u, v, w*; и где *x* – четвёртая переменная, по которой производится дифференцирование.

Cначала корректируются веса и пороги для последнего нейронного слоя по формуле. Затем, используя скорректированные веса и пороги последнего слоя, корректируются веса и пороги предпоследнего слоя и т.д. Корректируются веса и пороги слоёв в обратном порядке, т.е. в направлении от выхода ко входу сети. При этом для коррекции следующего (в обратном порядке) слоя используются уже скорректированные параметры из предыдущих слоёв. Таким образом выполняется шаг 4 алгоритма.

При нахождении минимума функции потерь есть вероятность попасть в локальный минимум вместо глобального минимума. В этом случае применяются методы выхода из локальных минимумов.

При корректировке весов и порогов по формуле градиент может стать слишком маленьким, что приведёт к замедлению или даже полной остановке обучения. До 2006 года проблема исчезающего градиента (the vanishing gradient problem) была основной причиной, из-за которой было невозможно обучать глубокие сети с множеством слоёв, потому что от слоя к слою градиент уменьшался при применении метода обратного распространения ошибки. Проблема исчезающего градиента вынуждала исследователей использовать неглубокие сети с небольшим количеством слоёв.

Проблема исчезающего градиента возникает из-за того, что градиент некоторого слоя является произведением градиентов всех предыдущих слоёв. Если градиенты на предыдущих слоях меньше единицы и лежат на отрезке [0, 1], тогда их произведение будет давать меньшее число. Например, 0,5 \* 0,5 = 0,25, а при многократном умножении чисел, меньших единицы, их произведение будет стремиться к нулю и становиться исчезающе малым: 0,510 ≈ 0,00098.

Градиенты в глубоких ИНС являются нестабильными и стремятся либо к нулю, либо к бесконечности при продвижении к первым слоям многослойной сети. Эта нестабильность является фундаментальной проблемой для методов обучения глубоких ИНС с помощью градиентов.

Переобучение (overfitting) – явление, при котором ошибка модели на объектах, не участвовавших в обучении, оказывается существенно выше, чем ошибка на объектах, участвовавших в обучении. Переобучение возникает при использовании слишком сложных моделей, как правило, с большим количеством нейронов и синапсов, либо при слишком долгом процессе обучения, либо при неудачной обучающей выборке.

Недообучение (underfitting) – явление, при котором ошибка обученной модели оказывается слишком большой. Недообучение возникает при использовании слишком простых моделей, как правило, с малым количеством нейронов и синапсов, либо при прекращении процесса обучения до достижения состояния с достаточно малой ошибкой, либо при неудачной обучающей выборке.

На рисунке 1.8 приведены примеры недообученной, обученной и переобученной модели в задаче регрессии.

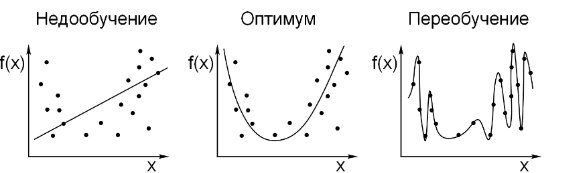


Рисунок 1.8 – Недообучение, оптимум и переобучение в регрессии

Причины переобучения разнообразные:

* слишком мало данных для обучения либо слишком много нейронов и синапсов в ИНС – модель запомнила все варианты из обучающей выборки и, таким образом, утратила возможность обобщения, выдавая запомненные варианты вместо предсказаний;
* слишком долгое обучение – модель находит закономерности в шуме (галлюцинирует);
* плохо подготовленные данные в обучающей выборке могут привести к тому, что модель будет давать большую ошибку на новых данных, которые не участвовали в обучении.

Если модель запомнила все варианты в обучающей выборке, то новые примеры она может не угадать просто потому, что они отличаются от тех, что были в выборке. Эта проблема возникает, когда обучающая выборка слишком маленькая либо модель ИНС слишком сложная.

Если модель находит закономерности в шуме, то новые данные будут обладать другим шумом, который всегда есть в данных, тогда ответ модели будет ошибочным. Эта проблема возникает, когда обучение было слишком долгим и модель просто подогнана под шум в обучающей выборке.

Для того, чтобы контролировать модель на переобучение, нужно использовать отдельные наборы данных для обучения и оценки. Данные, на которых происходит оценка качества, не должны участвовать в обучении.

1. **РАЗРАБОТКА ПРОГРАММНОГО КОДА**
   1. **Сбор данных**

Первым шагом для машинного обучения нужно собрать выборку данных, так как машинное обучение требует входные данные, в данном случае данные о подержанных автомобилях.

Было принято решение, что сбор данных будет осуществляться с сайта Avito.ru. Это решение обосновано несколькими причинами:

1. Популярность и обширная база объявлений: Avito.ru является одним из наиболее популярных и крупных онлайн-площадок в России для размещения объявлений о продаже различных товаров, включая автомобили. Благодаря своей популярности, на Avito.ru можно найти обширную базу объявлений о продаже подержанных автомобилей, что обеспечивает больший объем данных для анализа.
2. Разнообразие представленных марок и моделей: на Avito.ru представлены автомобили различных марок, моделей и конфигураций, что позволяет получить разнообразную и полную информацию о стоимости подержанных автомобилей на рынке.
3. Доступность для веб-скрапинга и API: Avito.ru обеспечивает открытый доступ к своим веб-страницам, что позволяет использовать метод веб-скрапинга для сбора данных. Кроме того, в некоторых случаях Avito.ru может предоставлять API для доступа к определенным данным, что дополнительно облегчает процесс сбора информации.
4. Региональное покрытие: Avito.ru активно используется не только в крупных городах, но и в регионах России, что позволяет получить данные о стоимости подержанных автомобилей из различных регионов страны, что может быть важно при анализе региональных рынков.

Avito.ru не предоставляет пользователям в открытом доступе свои файлы с данными, предположим, что это связано с конфиденциальностью платформы, следовательно этот вариант отпадает. Специализированные сервисы по сбору данных, а также базы данных в открытом доступе нам тоже не подходят, в силу задачи реализации алгоритма и программного модуля по сбору данных, а также в связи с их маловероятной актуальностью. Остается только веб-скрапинг и использование API сайта.

Хотя Avito и предоставляет API для доступа к данным о стоимости подержанных автомобилей, его использование осложнено несколькими факторами. Во-первых, доступ к API предоставляется только разработчикам с уже готовыми приложениями, что создает дополнительные сложности для исследователей, не имеющих готового программного обеспечения. Во-вторых, даже при наличии готового приложения процесс согласования с Avito может занять значительное количество времени из-за необходимости выполнения различных формальностей и прохождения процедур авторизации.

Учитывая ограничения, связанные с использованием API, а также невозможность доступа к файлам данных Avito, веб-скрапинг остается наиболее подходящим и эффективным способом сбора данных. Веб-скрапинг позволяет автоматизированно извлекать информацию непосредственно с веб-сайта, обходя ограничения, связанные с API, и не требует предварительного согласования с платформой. Такой подход обеспечивает более гибкий и быстрый доступ к данным, что особенно важно для процесса анализа и интерпретации информации о стоимости подержанных автомобилей на Avito.ru.

В выборе Python и Selenium для реализации веб-скрапинга на Avito.ru существует несколько весомых причин:

1. Универсальность и популярность Python. Python – один из самых популярных языков программирования в мире, известный своей универсальностью и широким сообществом. Его простой и читаемый синтаксис делает Python отличным выбором для начинающих, а также для опытных разработчиков. В контексте веб-скрапинга это означает легкость в изучении и применении, а также доступ к обширной базе знаний и готовым решениям для решения типичных задач.
2. Мощные библиотеки для веб-скрапинга. Python обладает мощными библиотеками, специализированными на веб-скрапинг, включая BeautifulSoup и Selenium. Эти инструменты предоставляют гибкие и мощные средства для извлечения данных из веб-страниц, обработки HTML и управления веб-браузерами. В то время как BeautifulSoup отлично подходит для работы со статическим контентом, Selenium позволяет взаимодействовать с динамически загружаемым контентом, что часто встречается на современных веб-сайтах.
3. Взаимодействие с динамическим контентом через Selenium. Многие веб-страницы, включая Avito, используют JavaScript для динамической подгрузки контента, что усложняет задачу веб-скрапинга. Selenium решает эту проблему, имитируя действия обычного пользователя: переход по ссылкам, нажатие на кнопки и ожидание загрузки контента. Это позволяет извлекать данные, которые не доступны при использовании только статического анализа HTML [13].
4. Гибкость и масштабируемость. Использование Python в сочетании с Selenium дает возможность гибко настраивать процесс сбора данных под конкретные нужды исследования. Это включает в себя возможность автоматизации входа в учетные записи, обход капчи и работы с cookies. Благодаря поддержке многопоточности и асинхронного программирования Python позволяет масштабировать процесс сбора данных, обрабатывая большие объемы информации за короткое время.
5. Открытость и доступность. Python является бесплатным и открытым инструментом, что делает его доступным для широкого круга исследователей и разработчиков. Вместе с Selenium, который также является открытым программным обеспечением, они формируют мощный и экономически выгодный набор инструментов для веб-скрапинга.

Сбор данных о продаже подержанных автомобилей будет осуществляться в городе Набережные Челны и его окружности радиусом 200 километров. Будут собираться следующие параметры:

1. Марка автомобиля
2. Модель автомобиля
3. Поколение
4. Конфигурация
5. Комплектация
6. Цена
7. Год выпуска
8. Пройденный пробег
9. Цвет
10. Расположение руля
11. Количество владельцев
12. Состояние автомобиля
13. Тип двигателя
14. Количество лошадиных сил
15. Тип кузова
16. Тип коробки
17. Привод

Этапы разработки алгоритма:

1. Подготовка рабочего окружения:
   1. Установка необходимых библиотек (selenium, selenium-stealth) через менеджер пакетов pip.
   2. Настройка веб-драйвера Chrome и опций для обхода детекции автоматического управления.
2. Создание структуры для хранения данных:
   1. Определение структуры данных (в данном случае, CSV-файл) для сохранения результатов скрапинга.
   2. Создание файла с заданными заголовками столбцов, соответствующими атрибутам автомобилей.
3. Разработка механизма сбора ссылок на объявления:
   1. Переход по страницам сайта Avito в разделе подержанных автомобилей.
   2. Извлечение и сохранение уникальных URL объявлений.
4. Скрапинг данных из объявлений:
   1. Переход по сохраненным ссылкам и сбор информации о характеристиках автомобилей, таких как марка, модель, год выпуска, пробег, цена и т.д.
   2. Обработка возможных исключений (NoSuchElementException) для устойчивости алгоритма.
5. Сохранение данных:
   1. Запись собранной информации в CSV-файл для последующего анализа.
6. Завершение работы скрапера:
   1. Корректное закрытие веб-драйвера после выполнения сбора данных.

Особенности и сложности:

1. Обход защиты от ботов: Использование selenium-stealth для имитации действий реального пользователя и обхода систем детекции автоматического управления браузером.
2. Управление временем ожидания: Необходимость добавления задержек (time.sleep) для имитации реального поведения пользователя и предотвращения блокировки со стороны сайта.
3. Обработка исключений: Важность обработки исключений для обеспечения устойчивости алгоритма к ошибкам в процессе скрапинга.

Разработка алгоритма.

Алгоритм работы парсера представлен на рисунке 2.1.

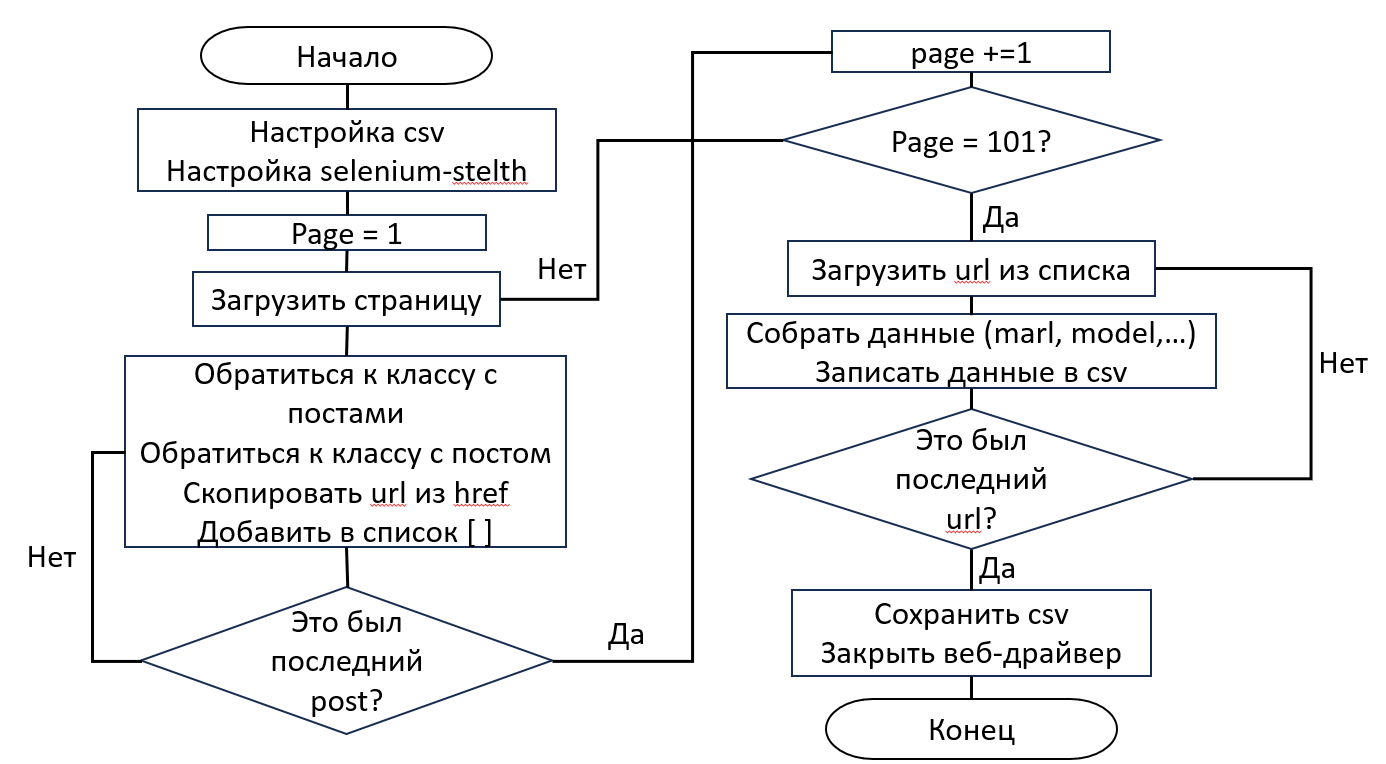


Рисунок 2.1 – Блок-схема работы парсера

Рассмотрим подробнее структуру программы:

1. Настройка веб-драйвера для обхода систем обнаружения.

В начале работы над алгоритмом парсинга, основным этапом является подготовка и настройка инструментов для обеспечения необнаруживаемого взаимодействия с целевыми веб-сайтами. Для этого используется специализированный инструмент — веб-драйвер Chrome, настроенный с поддержкой библиотеки selenium-stealth. Эта конфигурация позволяет имитировать действия обычного пользователя, минимизируя риск автоматического обнаружения парсинга. Настройки включают максимизацию окна браузера, отключение ряда флагов, связанных с автоматизацией, и подмену информации о браузере и операционной системе.

1. Сбор ссылок на объявления.

Далее следует этап сбора ссылок. Алгоритм автоматически перебирает страницы с объявлениями на выбранном сайте, используя в качестве примера сайт автомобильной тематики. Для каждой страницы формируется уникальный URL, включающий номер страницы и параметры фильтрации. Переход по страницам осуществляется с задержками, чтобы имитировать поведение пользователя. Из HTML-кода страниц извлекаются ссылки на конкретные объявления, которые сохраняются для последующего анализа. Важно отметить, что используются только уникальные URL, чтобы избежать повторной обработки.

1. Извлечение данных из объявлений.

На следующем этапе алгоритм переходит по каждой собранной ссылке для детального анализа содержимого объявления. Это включает в себя сбор информации о марке и модели автомобиля, а также различных характеристиках, таких как год выпуска, пробег, тип кузова, и др. Данные извлекаются из специализированных разделов или таблиц на странице. Помимо технических характеристик, алгоритм фиксирует стоимость автомобиля, которая часто выделена в отдельном элементе. Собранная информация систематизируется для дальнейшего использования.

1. Сохранение данных.

Собранные данные компилируются и записываются в файл формата CSV, который предварительно структурирован с определёнными заголовками. Это позволяет организовать информацию в удобном для анализа виде, обеспечивая легкость доступа и обработки данных.

1. Закрытие драйвера.

После завершения сбора и записи данных, алгоритм корректно завершает сессию работы с веб-драйвером, освобождая ресурсы и закрывая браузер. Это важный этап, обеспечивающий стабильность работы парсера и предотвращение утечек памяти.

Для обеспечения полноты представления материала и возможности практического применения разработанных методов, полный код алгоритма включён в состав работы как Приложение А.

В результате работы данного алгоритма всего было собрано 40426 строк данных о подержанных автомобилях. В каждой строке содержится информация о семнадцати признаках автомобиля. Эти признаки можно разделить на два типа:

1. Категориальные признаки – признаки, которые принимают ограниченное число возможных значений. Эти значения обычно представляют собой различные категории или классы.
2. Вещественные признаки (или числовые признаки) – признаки, которые принимают числовые значения. Эти значения могут представлять собой любое действительное число.

В таблице 2.1 представлена информация к какому типу относится признак в собранном наборе данных и его обозначение.

Таблица 2.1 – Типы признаков

|  |  |  |
| --- | --- | --- |
| Название признака | Обозначение признака | Тип признака |
| Mark | Марка автомобиля | Категориальный |
| Model | Модель автомобиля | Категориальный |
| Generation | Поколение | Категориальный |
| Configuration | Конфигурация | Категориальный |
| Complectation | Комплектация | Категориальный |
| Price\_rub | Цена | Вещественный |
| Year | Год выпуска | Вещественный |
| Km\_age | Пройденный пробег | Вещественный |
| Color | Цвет | Категориальный |
| Wheel | Расположение руля | Категориальный |
| Owners\_count | Количество владельцев | Вещественный |
| condition | Состояние | Категориальный |
| Engine\_type | Тип двигателя | Категориальный |
| Horse\_power | Кол-во лошадиных сил | Вещественный |
| Body\_type | Тип кузова | Категориальный |
| transmission | Тип коробки | Категориальный |
| Drive\_type | Привод | Категориальный |

Таким образом, разработанный алгоритм для сбора данных обладает необходимой гибкостью и масштабируемостью для адаптации к различным задачам веб-скрапинга. Благодаря использованию современных инструментов и технологий, а также соблюдению лучших практик программирования и этики веб-скрапинга, алгоритм способен эффективно собирать данные с сайтов объявлений, обеспечивая высокое качество и актуальность собранных данных для последующего анализа.

Применение такого подхода позволяет автоматизировать сбор больших объемов данных о стоимости подержанных автомобилей, сокращая время и усилия, необходимые для ручного сбора информации, и создавая надежную основу для аналитических исследований в данной области.

* 1. **Первичная обработка данных**

Первичная обработка данных перед машинным обучением – это критически важный этап, который напрямую влияет на качество и точность прогнозирования. Обучение на предварительно необработанных данных, может исказить результаты модели и значительно снизить точность прогноза [3].

В первую очередь требуется удалить строки, в которых значения всех переменных, кроме цены, пропущены. Возможно, при обработке данных объявлений парсером произошла какая-то ошибка. Для этого следует обратиться к признаку model и использовать метод isna() из библиотеки pandas [9].

Обработка дубликатов.

На первом этапе обработки данных проводится поиск и удаление дубликатов, то есть полностью идентичных записей. Это важно для обеспечения уникальности каждой записи в датасете. При анализе данных было выявлено, что в наборе данных присутствует 35 таких повторяющихся записей. Их удаление необходимо для повышения качества и надежности последующего анализа. Удалить такие записи можно использую метод drop\_duplicates().

* 1. Обработка пропущенных значений.

Следующий шаг — обработка пропущенных значений. Для этого вначале проводится анализ данных на предмет наличия незаполненных полей в таких важных атрибутах, как конфигурация автомобиля, комплектация, тип двигателя, мощность, тип кузова, тип трансмиссии и тип привода. Для заполнения пропущенных значений используется подход, основанный на замене пропусков наиболее часто встречающимися значениями (модой) для категориальных переменных и средним значением для числовых переменных. Этот метод позволяет сохранить структурную целостность данных и сделать датасет более полным для анализа. Информация по пропускам представлена на рисунке 2.2.

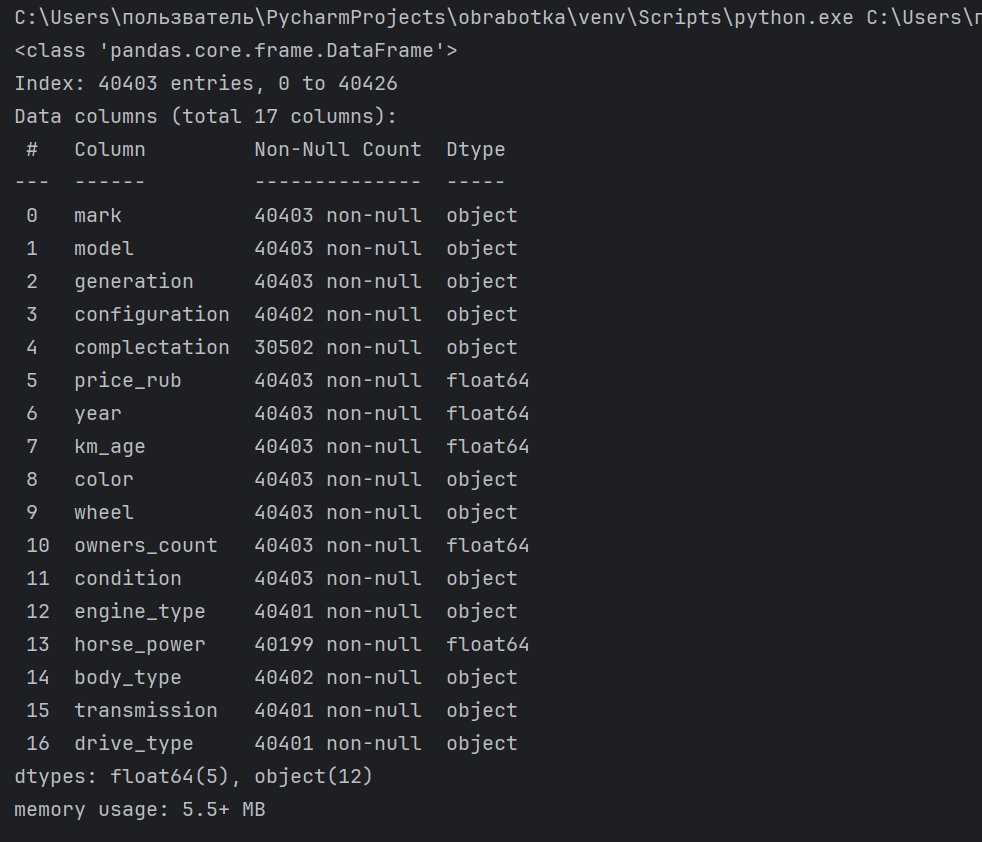


Рисунок 2.2 – Информация по датасету

Имеем следующие признаки с пропущенными значениями:

1. Configuration
2. Complectation
3. Engine\_type
4. Horse\_power
5. Body\_type
6. Transmission
7. Drive\_type

Заменить все пропущенные значения модой можно используя метод mean().

* 1. Преобразование переменных.

Далее происходит процесс преобразования типов переменных для упрощения их обработки и анализа. Например, цена автомобиля приводится к целочисленному формату для удобства расчетов. При этом из значений удаляются все нецифровые символы.

Обратим внимание на категориальный признак generation (рисунок 2.3).

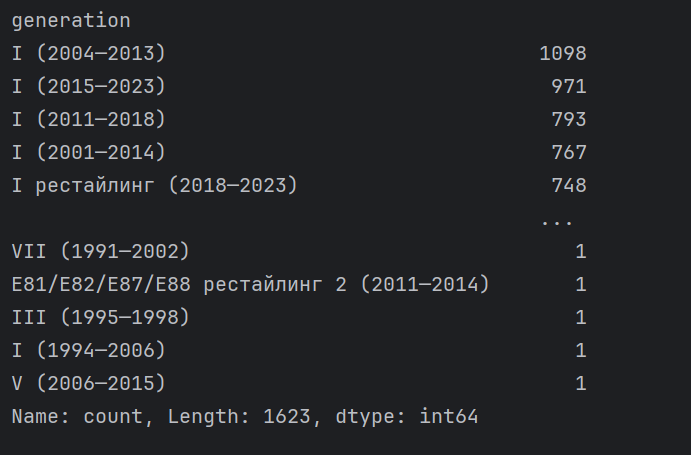


Рисунок 2.3 – Информация по признаку ‘generation’

1623 уникальных значений - слишком много, так как нужно учитывать, что в дальнейшем может понадобиться преобразовывать этот признак в вещественный и для этого потребуется множество столбцов для обозначения каждого уникального значения этого признака. К тому же, основное различие в признаках составляют года, указанные в скобках. Эта информация избыточна, поэтому нужно преобразовать признак. Например добавить бинарный признак restyling.

Уникальных значений стало 1458, теперь следует отбросить информацию в скобках и оставить только порядковый номер поколения, результаты можно увидеть на рисунке 2.4.

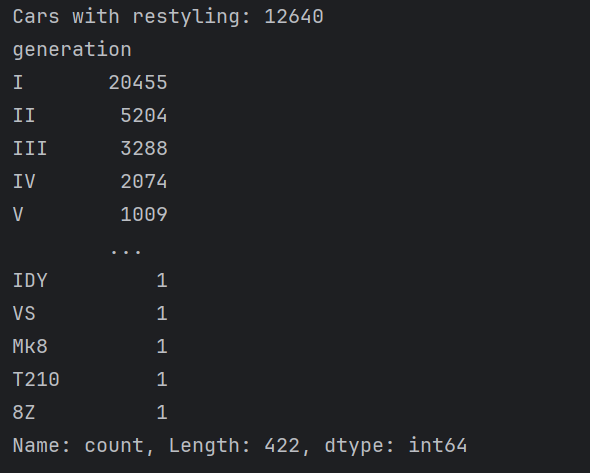


Рисунок 2.4 – Результат модификации значения ‘generation’

Рассмотрим признак configuration, рисунок 2.5.

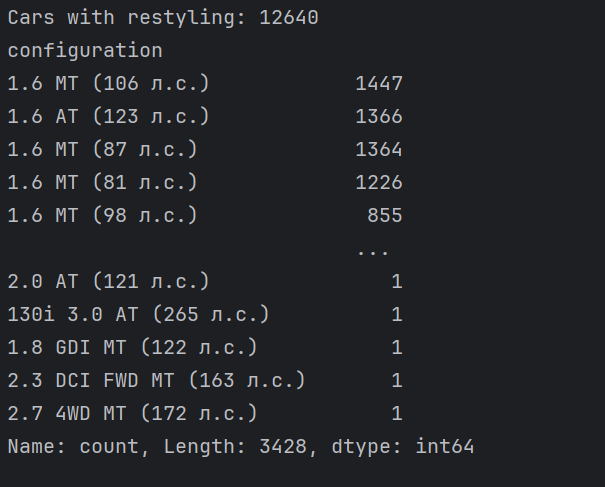


Рисунок 2.5 – Информация по признаку ‘configuration’

Имеем 3428 уникальных значений. Заметим, что в модификации содержится избыточная информация о коробке передач (MT - механическая, AT - автомат) и лошадиных силах. Извлечем из этого признака информация об объеме двигателя в новый признак engine\_volume и удалим configuration.

Посмотрим на признак complectation, рисунок 2.6.

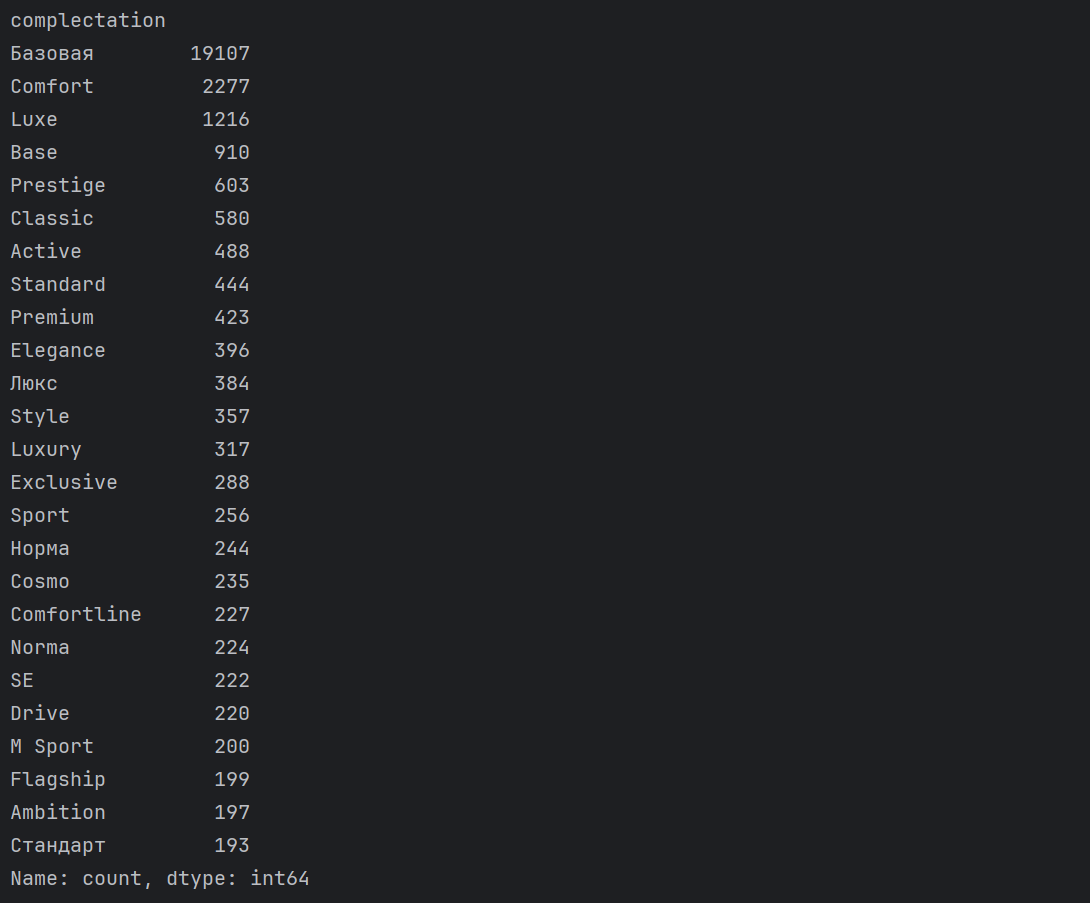


Рисунок 2.6 – Информация по признаку ‘complectation’

Видно, что многие уникальные значения означают одно и то же, только записаны по-разному (Norma и Норма, например). Заменим некоторые значения, создав список, где к каждому по-разному записанному значению будет присвоен ключ – какое-то одно из этих значений.

* 1. Выявление аномалий.

Особое внимание уделяется выявлению и обработке аномальных значений, таких как необычно низкая или высокая цена, а также крайне маленький или большой пробег автомобиля. Для этого применяются методы фильтрации данных, исключающие из анализа нереалистичные записи. Это позволяет повысить точность и надежность результатов исследования.

На рисунке 2.7 представлена статистика вещественных признаков.

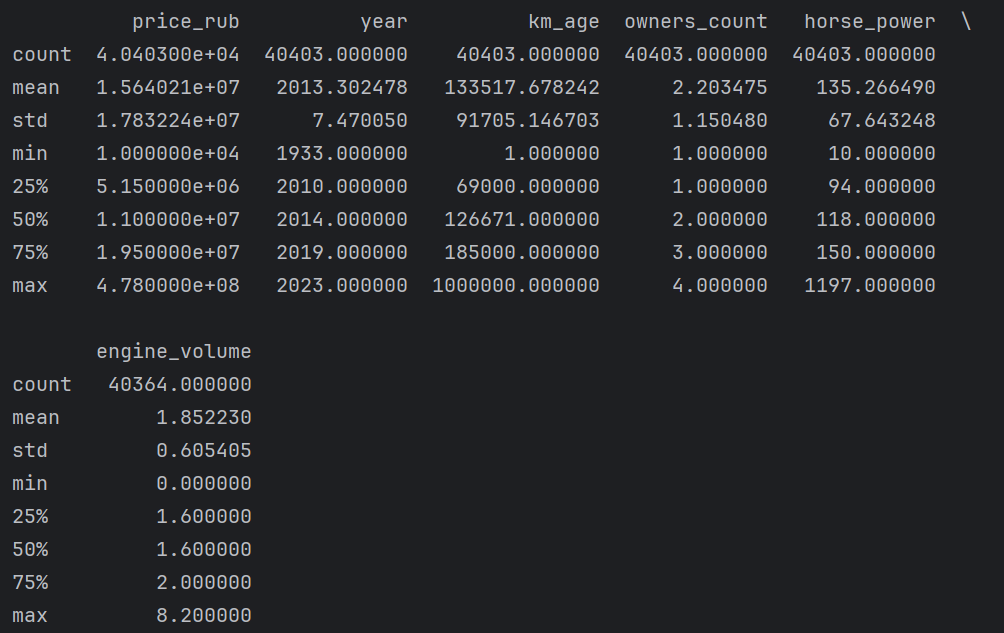


Рисунок 2.7 – Статистика вещественных признаков

Из таблицы видно, что в датасете присутствует аномальное объявление с ценой "10.000". Для реалий сегодняшнего рынка даже самый потрепанный автомобиль вряд ли будет стоить столь мало.

Также максимальное значение пробега 1.000.000 км выглядит подозрительно, как и пробег 1 км, нужно удалить строки с данными значениями.

По завершении всех этапов обработки и подготовки данных, итоговый набор данных сохраняется в файл. Сохраним очищенные и подготовленные данные в файл preprocessed\_data.csv.

После первичной обработки данных, в датасете осталось 36508 строк с данными и 18 признаков, а именно одна целевая переменная price\_rub, она будет прогнозироваться по результатам обучения, двенадцать категориальных признаков и пять вещественных. Такие данные можно использовать далее для машинного обучения.

Полный алгоритм первичного анализа данных в виде кода на python представлен в приложении Б.

* 1. **Программная реализация методов машинного обучения**

Так как большинство методов машинного обучения умеют работать только с вещественными данными, то следует сначала преобразовать все категориальные признаки в вещественные.

Категориальные признаки представляют собой переменные, которые принимают ограниченное число категорий или классов. Примерами могут служить цвет автомобиля, тип кузова, марка или модель. Непосредственно в их исходном виде большинство алгоритмов машинного обучения не могут эффективно работать с такими данными, так как они предназначены для обработки числовых входов.

Для трансформации категориальных признаков в числовые существует несколько подходов:

1. Кодирование с помощью меток (Label Encoding): каждая категория заменяется уникальным числом. Этот метод прост в реализации и не увеличивает размерность данных, но может ввести ложное представление о порядке (например, одна категория больше другой).
2. Прямое кодирование (One-Hot Encoding): каждая категория представляется собственным столбцом, где присутствие категории отмечается 1, а её отсутствие – 0. Этот метод устраняет ложное представление о численном порядке категорий, однако может значительно увеличить размерность данных, что затрудняет обучение моделей.

В рамках данной работы был выбран метод прямого кодирования для преобразования категориальных признаков. Решение обусловлено следующими факторами:

1. Необходимостью избежать любого предположения о порядке или весе различных категорий.
2. Способностью метода адаптироваться к моделям, которые требуют нумерического ввода без введения искажений в данные.

Преобразование было выполнено с использованием библиотеки scikit-learn и её компонента OneHotEncoder. После преобразования категориальных признаков, данные были объединены с числовыми признаками, что позволило подготовить окончательный набор данных для обучения моделей.

Теперь давайте перейдем к подготовке числовых признаков. Этот процесс также крайне важен и требует внимательного рассмотрения.

Числовые признаки, такие как мощность двигателя, год выпуска автомобиля, пробег, количество владельцев и объем двигателя, играют ключевую роль в многих алгоритмах машинного обучения. Однако различия в масштабах и распределениях этих признаков могут привести к проблемам при обучении моделей. Для устранения этих проблем применяется масштабирование признаков.

Масштабирование необходимо для нескольких целей:

1. Улучшение сходимости алгоритмов: многие алгоритмы машинного обучения, особенно те, что используют методы градиентного спуска (например, большинство нейронных сетей), работают лучше, когда все входные признаки находятся примерно в одном и том же масштабе.
2. Уменьшение влияния выбросов: масштабирование также может помочь уменьшить влияние выбросов на процесс обучения, делая модель более устойчивой.
3. Обеспечение равной важности признаков: без масштабирования признаки с большими абсолютными значениями могут оказывать большее влияние на модель, что не всегда желательно.

Существуют различные методы масштабирования, включая:

1. Стандартизация (Z-оценка): преобразует признаки так, чтобы их среднее значение было равно 0, а стандартное отклонение — 1. Этот метод удобен тем, что он обрабатывает данные так, чтобы они соответствовали стандартному нормальному распределению.
2. Нормализация (Min-Max Scaling): масштабирует данные к заданному диапазону, обычно от 0 до 1. Этот метод полезен, когда нужно сохранить нулевые значения в данных или когда признаки не распределены нормально [23].

В рамках данной работы была выбрана, как и стандартизация, так и нормализация числовых признаков с использованием класса StandardScaler из библиотеки scikit-learn.

Для прогнозирования цены в выборку был добавлен входной признак (столбец с обозначением категории), все данные были поделены на категории с разницей в 150000 рублей, в результате получилось 50 категорий. Количество категорий определялось опытным путем. Изначально данные были поделены на 5 категорий, потом число категорий постепенно увеличивалось, таким образом было определено, что оптимальное количество категорий равно 50.

Далее в выборку был добавлен столбец с нормализованной ценой, то есть цена каждого автомобиля в выборке делилась на максимальную цену категории этого автомобиля и получалось значение от 0 до 1, прогнозирование будет происходить по нормализированной цене и потом просто умножаться на максимальную цену категории, чтобы на выходе была спрогнозированная цена в рублях.

Следующий этап это перемешивание набора данных и разделение их на обучающие и тестовые, использовалось стандартное разделение для машинного обучения – 70 на 30, то есть 70 процентов данных из выборки ушло на обучение моделей, а 30 процентов на тестирование.

Далее идет обучение моделей и их оценивание на тестовом наборе, но перед этим стоит уточнить, что я подумал, будет глупо если пользователь сам будет вводить к какой категории относится его автомобиль, так как цель программы – прогнозирование стоимости, а значит стоит учитывать, что пользователь может даже примерно не знать стоимость своего автомобиля. Тогда было принято решение – прогнозировать не только стоимость, но перед этим еще и категорию. То есть в дальнейшем пользователь будет вводить только характеристики автомобиля, а дальше программа уже сама будет прогнозировать сначала категорию, а потом уже и саму стоимость автомобиля. Это означает, что помимо задачи регрессии, добавляется еще и задача классификации.

Прежде чем приступить к обучению моделей машинного обучения для решения задачи регрессии, следует сначала решить задачу классификации. Было принято решение, что для классификации будет использоваться только одна модель машинного обучения – искусственная нейронная сеть. Рассмотрим подробнее, почему в данном случае, при решении задачи классификации лучше, отказаться от использования линейной регрессии, лассо регрессии, экстремального градиентного бустинга и случайного леса решений в пользу искусственных нейронных сетей.

Линейная регрессия:

* Природа задачи: линейная регрессия изначально предназначена для решения задач регрессии, а не классификации. Она предсказывает непрерывные значения, тогда как задача классификации требует предсказания категорий.
* Модель: для применения линейной регрессии к классификации (например, через пороговое значение) необходимо адаптировать модель, что не всегда эффективно и может приводить к снижению точности.
* Линейность: линейная регрессия предполагает линейную зависимость между признаками и откликом. Это ограничение приводит к неадекватным результатам, если данные имеют нелинейные зависимости.

Лассо регрессия:

* Природа задачи: подобно лассо регрессии, лассо регрессия изначально предназначена для регрессии. Лассо добавляет регуляризацию, что полезно для отбора признаков, но не решает проблему принципиально.
* Сложные зависимости: лассо регрессия также предполагает линейные зависимости, что ограничивает её применимость в задачах с нелинейной природой.

Экстремальный градиентный бустинг (XGBoost):

* Настройки и сложность: XGBoost является мощным и сложным алгоритмом с множеством гиперпараметров. Настройка модели требует значительного времени и вычислительных ресурсов, что может быть неоправданным при наличии более простых и эффективных альтернатив.
* Обработка данных: XGBoost менее устойчив к отсутствующим значениям и аномалиям в данных по сравнению с нейронными сетями, которые могут включать механизмы, такие как регуляризация и различные виды слоев для обработки таких случаев.

Случайный лес решений:

* Скорость и масштабируемость: случайный лес может быть медленным при обучении и предсказании, особенно на больших объемах данных. При этом потребление памяти также значительно возрастает.
* Нелинейность: хотя случайный лес хорошо справляется с нелинейностями, он может уступать нейронным сетям в задачах с очень сложными нелинейными зависимостями, так как нейронные сети могут учиться более сложным функциям благодаря многослойной структуре.

Преимущества искусственной нейронной сети:

* Гибкость: нейронные сети могут аппроксимировать любую сложную функцию, что делает их очень мощными для задач с нелинейными и сложными зависимостями.
* Обработка большого объема данных: нейронные сети хорошо масштабируются и эффективно работают с большими объемами данных, используя методы, такие как батч-нормализация и методы оптимизации, которые ускоряют обучение.
* Многообразие архитектур: существует множество архитектур нейронных сетей, которые могут быть адаптированы под конкретные задачи (например, сверточные нейронные сети для изображений, рекуррентные нейронные сети для временных рядов и др.).
* Регуляризация: современные методы регуляризации, такие как dropout, batch normalization и другие, позволяют избежать переобучения и улучшить обобщающую способность модели.

Таким образом, использование искусственных нейронных сетей для задач классификации может быть предпочтительнее из-за их гибкости, способности работать с нелинейными зависимостями, обработки больших объемов данных и разнообразия архитектур, которые можно адаптировать под конкретные задачи. В то время как линейная и лассо регрессии ограничены своей линейной природой, XGBoost и случайный лес могут быть слишком сложными и ресурсоемкими.

Обучения искусственной нейронной сети для задачи классификации.

Для данной работы были использованы две искусственные нейронные сети, разработанные с использованием библиотеки TensorFlow Keras для решения задачи классификации и регрессии. Архитектура каждой модели включает в себя полносвязные слои прямого распространения.

Модель классификации цен автомобилей, архитектура модели включает:

1. Входной слой с размерностью, соответствующей количеству признаков в X\_train\_cat.
2. Первый скрытый слой из 128 нейронов с активацией ReLU.
3. Второй скрытый слой из 64 нейронов, также с активацией ReLU.
4. Третий скрытый слой из 32 нейронов с активацией ReLU.
5. Выходной слой с числом нейронов, равным количеству ценовых категорий, использует функцию активации softmax для классификации.

Модель обучается с использованием оптимизатора Adam и функции потерь sparse\_categorical\_crossentropy. В процессе обучения проводится валидация на 20% данных, а также используется EarlyStopping для предотвращения переобучения. Модель оценивается по точности (accuracy).

Точность (accuracy) — это одна из наиболее распространенных метрик для оценки качества моделей классификации. Она показывает долю правильно классифицированных объектов среди всех объектов в наборе данных. Точность является важным показателем эффективности в задачах классификации, особенно когда классы сбалансированы. Она рассчитывается по формуле:

где: *TP* (True Positive) – количество правильно определенных положительных случаев; *TN* (True Negative) – количество правильно определенных отрицательных случаев; *FP* (False Positive) – количество ошибочно классифицированных как положительные случаев; *FN* (False Negative) – количество ошибочно классифицированных как отрицательные случаев.

Точность измеряется в долях от 0 до 1 или в процентах от 0% до 100%. Чем выше значение точности, тем лучше модель справляется с классификацией по всем классам.

Оценка точности для модели классификации представлена на рисунке 2.8.

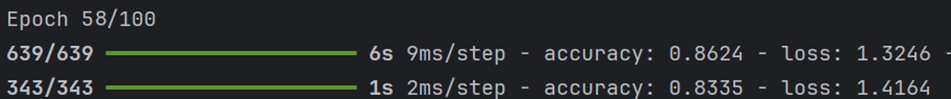


Рисунок 2.8 – Оценка точности ИНС модели классификации

По результатам оценки видно, что модель прогнозирует правильную классификацию в 83% случаях. Это не самые плохие, но и не самые хорошие результаты, в дальнейшем нужно будет посмотреть, как модель справляется на реальных данных.

Далее по порядку будут описаны все методы машинного обучения для решения задачи регрессии.

Линейная регрессия.

Построим модель линейной регрессии использую библиотеку sklearn, импортируем оттуда модуль Ridge, создадим объект этого класса и сразу же обучим его методом этого класса. Оценим точность обученной модели используя метрики точности, такие как MAE, R2, MSE на обучающих и тестовых данных. Результат можно увидеть на рисунке 2.9.

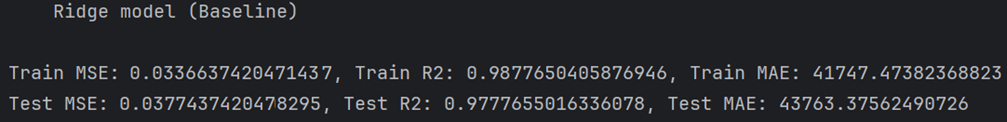


Рисунок 2.9 – Точность на обучающих и тестовых данных

Отсюда видно, что ошибка равна примерно 42000 рублей и ошибки на тренировочном и тестовом наборах почти совпадают, а это значит, что текущая модель обладает хорошей предсказательной способностью и отсутствием переобучения.

Лассо регрессия.

Для обучения модели Лассо регрессии была использована библиотека sklearn, из которой был импортирован модуль Lasso. Эта модель позволяет ввести L1 регуляризацию, которая способствует уменьшению величины и обнулению некоторых коэффициентов, что делает модель более интерпретируемой и предотвращает переобучение.

Один из ключевых гиперпараметров модели Лассо – параметр регуляризации «alpha». Он контролирует степень влияния штрафа за сложность модели, добавляя к функции потерь слагаемое, пропорциональное сумме абсолютных значений коэффициентов регрессии. Выбор оптимального значения «alpha» критичен, поскольку он напрямую влияет на баланс между смещением и дисперсией в модели, а также на количество признаков, которые модель окончательно использует.

Для систематического подбора этого параметра и определения его оптимального значения был применён метод GridSearchCV из sklearn, который позволяет выполнить кросс-валидацию для каждого значения из заданного диапазона. В данном случае, значение «alpha» было выбрано из равномерно распределённой последовательности между 1\*10−51\*10−5 (0.00001) и 1\*10−31\*10−3 (0.001), сгенерированной функцией np.linspace, которая создала 10 возможных значений для «alpha». Такой подход позволяет обеспечить тщательный и объективный выбор параметра, минимизируя риск переобучения и максимизируя точность предсказаний модели на новых данных. Результат оценки точности модели представлен на рисунке 2.10.

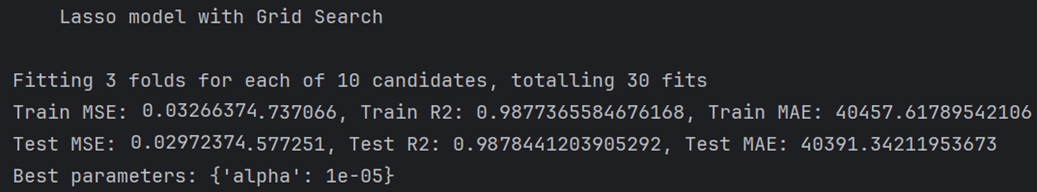


Рисунок 2.10 – Точность на обучающих и тестовых данных

Ошибка равна примерно 42000 рублей и ошибки на тренировочном и тестовом наборах почти идентичны, лучший параметр «alpha» равен

Случайный лес решений.

Для обучения модели Случайного леса была использована библиотека «sklearn», из которой был импортирован модуль «RandomForestRegressor». Эта модель ансамблевого обучения основана на множестве деревьев решений, работающих как один коллективный орган. Каждое дерево в Случайном лесу строится из случайно отобранной выборки обучающих данных, что делает модель устойчивой к переобучению и повышает её обобщающую способность.

Один из ключевых гиперпараметров модели Случайного леса – количество деревьев «n\_estimators». Этот параметр определяет число деревьев в лесу. Большее количество деревьев увеличивает точность предсказаний модели за счёт снижения дисперсии и уменьшения риска переобучения, но при этом увеличивается время обучения и вычислительные затраты.

Для выбора оптимального количества деревьев в лесу был использован метод «GridSearchCV» из «sklearn». Этот метод позволяет выполнить кросс-валидацию для каждого значения из заданного диапазона и определить, при каком количестве деревьев модель демонстрирует наилучшую производительность. В данном исследовании было решено использовать фиксированное значение «n\_estimators» равное 200, опираясь на предварительные анализы и экспертыные рекомендации, что обеспечивает хороший баланс между точностью предсказаний и вычислительной эффективностью. Оценки точности модели представлены на рисунке 2.11.

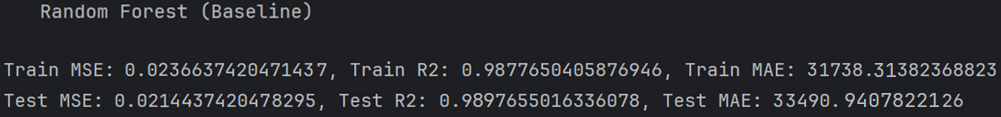


Рисунок 2.11 - Точность на обучающих и тестовых данных

Ошибка равна примерно 32000 рублей и ошибки на тренировочном и тестовом достаточно близки друг к другу.

Экстремальный градиентный бустинг.

Для обучения модели XGBoost была использована библиотека xgboost в Python, которая предоставляет эффективную реализацию алгоритма. Модель XGBoost включает в себя множество параметров, позволяющих тонко настраивать процесс обучения и оптимизации, что делает её исключительно гибкой. Рассмотрим подобранные параметры:

1. n\_estimators:6000  
   Этот параметр определяет количество деревьев, которые будут построены в процессе бустинга. Выбор большого количества деревьев может улучшить точность модели за счёт более детального анализа данных, но также увеличивает риск переобучения и вычислительные затраты. Значение 6000 указывает на стремление к высокой точности при достаточных вычислительных ресурсах.
2. Gamma:0  
   Параметр gamma устанавливает минимальное снижение потерь, необходимое для выполнения дополнительного разделения на листе дерева. Установка в 0 означает, что разделение листа будет выполняться всякий раз, когда это возможно, независимо от улучшения в точности, что делает каждое дерево более детализированным.
3. learning\_rate:0.03  
   Скорость обучения контролирует вклад каждого нового дерева в итоговую модель. Меньшее значение (0.03) замедляет процесс обучения, позволяя модели более тщательно адаптироваться под обучающие данные, потенциально увеличивая общую точность и уменьшая риск переобучения.
4. max\_depth:4  
   Максимальная глубина каждого дерева влияет на способность модели улавливать сложные закономерности в данных. Значение 4 предполагает, что модель будет достаточно сложной, чтобы выявлять нелинейные зависимости, но не слишком сложной для переобучения.
5. min\_child\_weight:2  
   Этот параметр определяет минимальное необходимое количество наблюдений (взвешенных), которое должно находиться в каждом листе. Значение 2 гарантирует, что каждое дерево будет достаточно обобщающим, предотвращая создание чрезмерно специфических правил на основе небольшого количества данных.
6. Objective:«reg:squarederror»  
   Определяет функцию потерь, которая будет минимизироваться при обучении. Для задач регрессии 'reg:squarederror' указывает на минимизацию среднеквадратичной ошибки, что является стандартным выбором для таких задач.
7. reg\_alpha**:**0.75…и…reg\_lambda**:**0.5  
   Эти параметры представляют собой коэффициенты L1 (Lasso регуляризация) и L2 (Ridge регуляризация) соответственно. Reg\_alpha равный 0.75 помогает уменьшить размерность модели за счёт обнуления некоторых коэффициентов. Reg\_lambda равный 0.5 уменьшает риск переобучения за счёт штрафа за слишком большие веса в модели.
8. subsample:0.8  
   Определяет долю обучающего набора, используемую для обучения каждого дерева. Значение 0.8 означает, что каждое дерево будет обучаться на 80% выборки, что помогает уменьшить переобучение и увеличивает робастность модели за счёт случайной выборки данных.

Каждый из этих параметров тщательно выбран, чтобы сбалансировать между точностью предсказаний и риском переобучения, а также оптимизировать вычислительные затраты и время обучения. Оценки точности можно увидеть на рисунке 2.12.

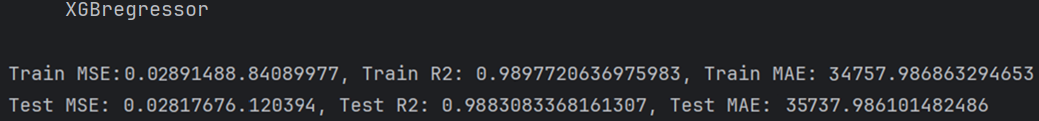


Рисунок 2.12 – Точность на обучающих и тестовых данных

Достаточно хорошие результаты, средняя ошибка на тестовой выборке равна примерно 35000 рублей, разница между обучающей и тестовой выборками 1020 рублей.

Искусственная нейронная сеть. Задача регрессии.

Модель предсказания цены автомобиля, архитектура модели содержит:

1. Входной слой, принимающий данные с признаками из X\_train\_price.
2. Первый скрытый слой с 128 нейронами и функцией активации ReLU.
3. Второй скрытый слой с 64 нейронами и активацией ReLU.
4. Третий скрытый слой из 32 нейронов с активацией ReLU.
5. Выходной слой с одним нейроном, предназначен для регрессии, не требует активационной функции, так как он должен выдавать непрерывные значения в диапазоне, которые могут быть любыми вещественными числами.

Обучение проводится с оптимизатором Adam и функцией потерь mean\_squared\_error. Применяется валидация на 20% тренировочных данных и EarlyStopping. Оценка модели происходит с использованием метрик среднеквадратичной ошибки и средней абсолютной ошибки (mae).

Оценки точности ИНС для модели регрессии показаны на рисунке 2.13.

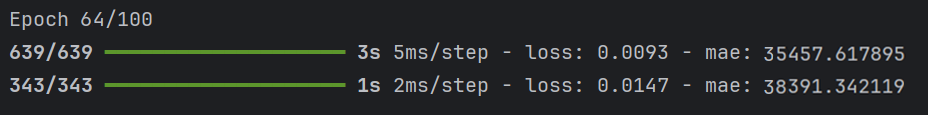


Рисунок 2.13 – Оценка точности ИНС модели регрессии

На тестовой выборке ИНС в среднем ошибалась на 38000 рублей.

Теперь следует проанализировать все полученные результаты по итогам обучения моделей. Построим таблицу с полученными точностями всех методов и сравним их.

Таблица 2.2 – Сравнение точности методов машинного обучения

|  |  |  |  |
| --- | --- | --- | --- |
|  | Средняя квадратичная ошибка | Коэффициент детерминации R2 | Средняя ошибка в руб. |
| Линейная регрессия | 0.03366 | 0.9877 | 43763 |
| Лассо регрессия | 0.02972 | 0.9878 | 40391 |
| Экстремальный градиентный бустинг | 0.02891 | 0.9883 | 35737 |
| Случайный лес | 0.02366 | 0.9877 | 33490 |
| Искусственная нейронная сеть | 0.02881 | 0.9881 | 38391 |

Из результатов всех методов машинного обучения, можно сделать вывод, что лучшая модель – случайный лес, а худшая – линейная регрессия. Обосновать результаты моделей можно следующим образом:

1. Дерево решений

Дерево решений выделилось как наиболее успешное благодаря своей способности улавливать сложные закономерности в данных без необходимости предполагать их распределение. Эта модель хорошо справляется как с категориальными, так и с числовыми данными, что типично для информации о б/у автомобилях. Но стоит учесть, что деревья решений могут переобучаться, если обучающие данные меняются.

1. Экстремальный градиентный бустинг (XGBoost)

XGBoost занял второе место по эффективности, строя серию деревьев решений, каждое из которых исправляет ошибки предыдущего. Это помогает снизить ошибку модели, но требует тщательной настройки параметров.

1. Искусственная нейронная сеть

Нейронные сети хорошо подходят для выявления сложных нелинейных зависимостей в больших объемах данных, но их эффективность сильно зависит от выбора архитектуры и глубины сети. Для данной работы нейронная сеть оказалась на третьем месте, вероятно из-за недостаточного количества данных или недостаточно точной настройки параметров, что не позволило полностью реализовать потенциал метода. Однако найти более точную настройку параметров, увы, не удалось.

1. Лассо регрессия

Лассо регрессия применяет L1-регуляризацию для уменьшения количества используемых признаков, что может быть полезно в ситуациях, когда некоторые переменные мало влияют на целевую переменную или между ними существует мультиколлинеарность. Это делает модель проще и часто более интерпретируемой, но может не обеспечить достаточной гибкости для сложных задач, в том числе таких как прогнозирование стоимости автомобилей.

1. Линейная регрессия

Линейная регрессия показала себя наименее эффективной, что может быть связано с её основным ограничением: предположением о линейной зависимости между переменными. В реальности взаимосвязи в данных о подержанных автомобилях могут быть значительно сложнее, включая нелинейные зависимости, которые линейная модель не может адекватно отразить.

Такое объяснение производительности различных моделей машинного обучения позволяет понять, почему одни подходы работают лучше других в конкретной задаче прогнозирования стоимости подержанных автомобилей.

Полный алгоритм машинного обучения представлен в приложении В.

* 1. **Реализация графического интерфейса пользователя**

В рамках разработки системы прогнозирования стоимости подержанных автомобилей ключевым аспектом является не только алгоритмическая составляющая, но и удобство взаимодействия пользователя с системой. С учетом современных тенденций и доступности цифровых технологий, особое внимание было уделено выбору интерфейса, который бы обеспечил максимальное удобство и оперативность использования разработанного решения. В результате было принято решение использовать графический интерфейс в виде Telegram-бота.

Telegram — это мессенджер, популярный во многих странах мира, который предоставляет обширные возможности для создания и настройки ботов. Использование Telegram-бота в качестве интерфейса для нашей системы прогнозирования имеет несколько значительных преимуществ. Прежде всего, это позволяет пользователю взаимодействовать с системой в привычной и доступной среде, не требуя установки дополнительных приложений или программ. Также Telegram обеспечивает высокую скорость передачи данных и мгновенное получение результатов, что является критически важным для задач, требующих оперативного решения [6].

Преимущества использования Telegram-бота:

1. Доступность и удобство: большинство пользователей уже знакомы с Telegram, что уменьшает кривую обучения и сопротивление при внедрении новой системы
2. Мгновенная обратная связь: бот может обрабатывать запросы в реальном времени, предоставляя быстрые ответы на вопросы пользователей
3. Кроссплатформенность: Telegram доступен на всех основных платформах, включая iOS, Android, Windows и macOS, что делает его универсальным решением
4. Безопасность: Telegram известен своими возможностями шифрования и защиты данных пользователей

Возможные трудности:

1. Ограничения API: Telegram имеет определенные ограничения по частоте и объему отправляемых сообщений, что может влиять на производительность при высокой нагрузке
2. Зависимость от стороннего сервиса: работа бота зависит от стабильности и доступности серверов Telegram
3. Взаимодействие с пользователем: необходимо тщательно продумать интерфейс взаимодействия, чтобы он был интуитивно понятен и максимально функционален

Для создания Telegram-бота на языке программирования Python доступно несколько библиотек, таких как python-telegram-bot, telepot, pyTelegramBotAPI. Каждая из этих библиотек имеет свои особенности, преимущества и недостатки. Однако, для реализации данного проекта была выбрана библиотека telebot.

Выбор telebot был обусловлен несколькими ключевыми факторами:

1. Простота использования: telebot предоставляет простой и интуитивно понятный интерфейс для взаимодействия с API Telegram. Это позволяет быстро создавать ботов с минимальным количеством кода
2. Богатая функциональность: Библиотека поддерживает все основные типы обновлений, которые предоставляет Telegram API, включая сообщения, команды, кнопки для быстрых ответов и встроенные запросы
3. Активное сообщество и поддержка: telebot имеет активное сообщество разработчиков и пользователей, которые регулярно обновляют библиотеку, исправляют ошибки и расширяют функциональность. Это обеспечивает легкость в решении возникающих вопросов и доступ к новым возможностям
4. Примеры и документация: telebot предлагает обширную документацию и множество примеров, что существенно упрощает процесс обучения и реализации специфических функций бота
5. Гибкость в настройке: Эта библиотека позволяет легко интегрировать различные виды обработчиков, что делает возможным тонкую настройку поведения бота в зависимости от потребностей проекта

Перед тем как приступить к программированию бота, необходимо создать самого бота в Telegram и настроить его базовые параметры. Этот процесс включает следующие шаги:

1. Создание бота через BotFather:
   1. В Telegram нужно найти пользователя @BotFather, который является официальным ботом для создания и управления другими ботами
   2. Написать ему /newbot и следовать инструкциям для создания нового бота. BotFather предложит ввести имя для бота и затем уникальное username, которое должно заканчиваться на ‘bot’ (например, example\_bot).
   3. После создания бота BotFather предоставит токен — уникальный ключ, который используется для управления ботом через API.
2. Настройка бота через BotFather:
   1. Можно установить описание, аватар и другие настройки бота, используя соответствующие команды (/setdescription, /setuserpic и др.).

В результате этих действий был создан бот с именем «AutoValue AI Predictor» и уникальным именем «Prognozcarbot». Ему был установлен аватар и добавлено описание.

Архитектура системы:

1. Импорт необходимых библиотек: В код включены библиотеки для работы с Telegram, обработки данных и выполнения машинного обучения.
2. Инициализация бота: Создание экземпляра бота с использованием токена аутентификации.
3. Загрузка моделей и объектов предварительной обработки: Система загружает предобученные модели машинного обучения и инструменты для предобработки данных.
4. Обработка команд от пользователя: Определение обработчиков сообщений для различных команд (/start, информационные запросы, запросы оценки автомобиля).
5. Сбор данных от пользователя: Интерактивный сбор данных об автомобиле через последовательность запросов.
6. Предобработка данных: Подготовка данных для модели с использованием методов кодирования и масштабирования.
7. Прогнозирование стоимости: Использование загруженных моделей для оценки стоимости автомобиля.
8. Вывод результатов: Отправка пользователю информации о прогнозируемой стоимости и ссылок на похожие объявления.
9. Обработка исключений: Управление ошибками и исключениями, возникшими в процессе работы бота.

Описание работы функций:

1. Функция preprocess\_input

Эта функция отвечает за предобработку данных пользователя перед подачей их в модели машинного обучения. Принимая DataFrame с данными пользователя, функция использует объекты onehot\_encoder и scaler для преобразования категориальных и числовых данных соответственно:

Категориальные данные преобразуются в one-hot вектора с помощью onehot\_encoder.

Числовые данные нормализуются с помощью scaler.

Преобразованные данные объединяются и возвращаются в формате, пригодном для использования в моделях машинного обучения.

1. Функция send\_welcome

Эта функция обрабатывает команду /start, отправляемую пользователем при первом взаимодействии с ботом. Она отправляет приветственное сообщение и предлагает пользователю три кнопки для выбора действия: оценка автомобиля, связь с администратором, и информация о боте. Это позволяет пользователю легко навигировать по функционалу бота.

1. Функция send\_main\_menu

Подобно функции send\_welcome, эта функция предоставляет пользователю главное меню в любой момент взаимодействия, позволяя вернуться к основным опциям после выполнения определенных задач или при возникновении ошибок.

1. Функция contact\_admin

Функция активируется, когда пользователь выбирает опцию "Связаться с администратором". Она отправляет сообщение с контактными данными администратора, обеспечивая возможность обратной связи.

1. Функция show\_info

При выборе опции "Информация" эта функция предоставляет детальную информацию о возможностях и ограничениях бота, помогая пользователю понять, чего ожидать от предсказаний цен.

1. Функция evaluate\_car

Инициирует процесс оценки автомобиля, вызывая функцию ask\_feature, которая начинает серию запросов для сбора данных об автомобиле.

1. Функция ask\_feature

Эта функция задает пользователю вопросы о характеристиках автомобиля, используя определенные кнопки для упрощения ввода данных. Она динамически изменяет вопросы в зависимости от предыдущих ответов пользователя.

1. Функция process\_feature

После получения каждого ответа от пользователя, process\_feature сохраняет эту информацию и переходит к следующему вопросу, пока не будут собраны все необходимые данные.

1. Функция predict\_price

После сбора всех данных функция создает DataFrame, выполняет их предобработку и использует модели машинного обучения для оценки стоимости. Результаты предсказаний представляются пользователю, а также предлагается ссылка на похожие объявления на сайте Avito, что дает дополнительный контекст и помощь в оценке цены.

Эти функции вместе формируют полноценное приложение Telegram-бота, которое обеспечивает удобный интерфейс для оценки стоимости подержанных автомобилей и предоставляет пользователю информацию, необходимую для обоснованных решений о покупке или продаже транспортного средства. Полный алгоритм разработанного графического интерфейса для пользователя в виде кода на python представлен в приложении Г.

Функционал разработанного интерфейса Telegram-бота можно увидеть на рисунках 2.14 – 2.17.



Рисунок 2.14 – Первое использование бота

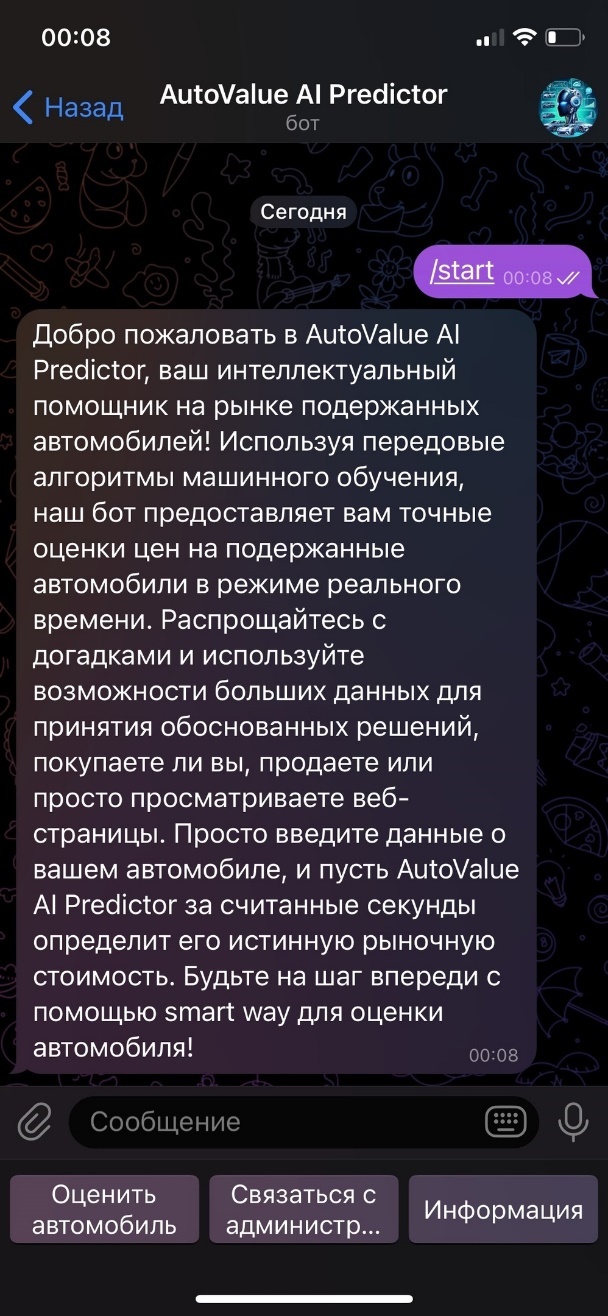


Рисунок 2.15 – Приветствие бота

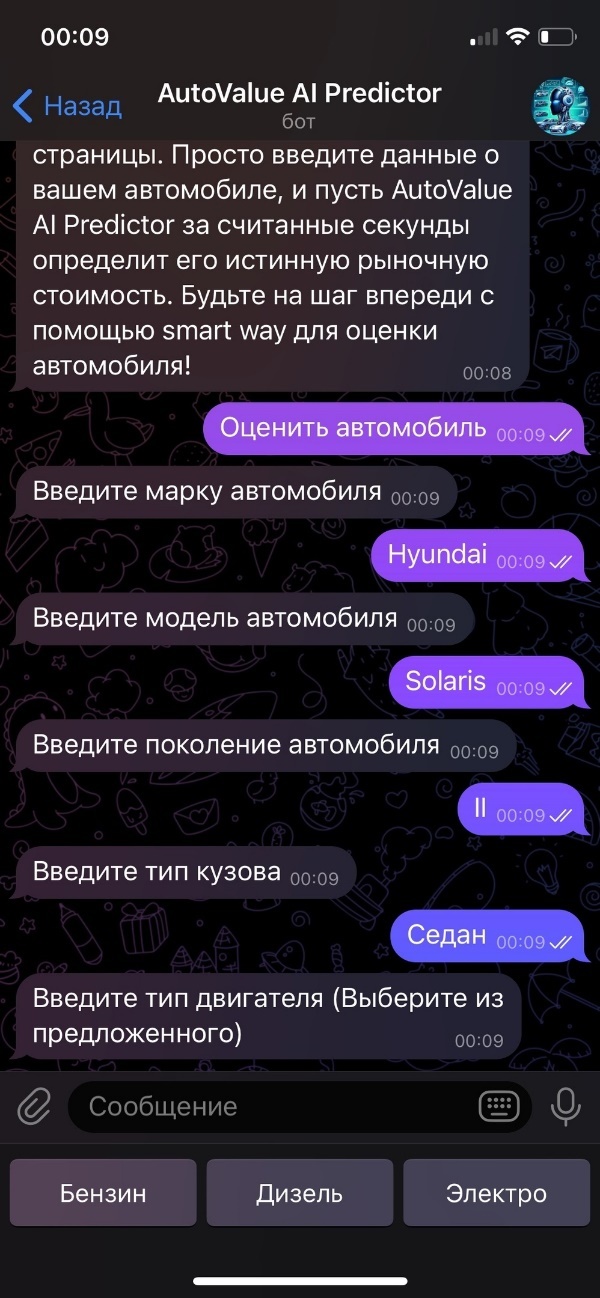


Рисунок 2.16 – Процесс работы бота

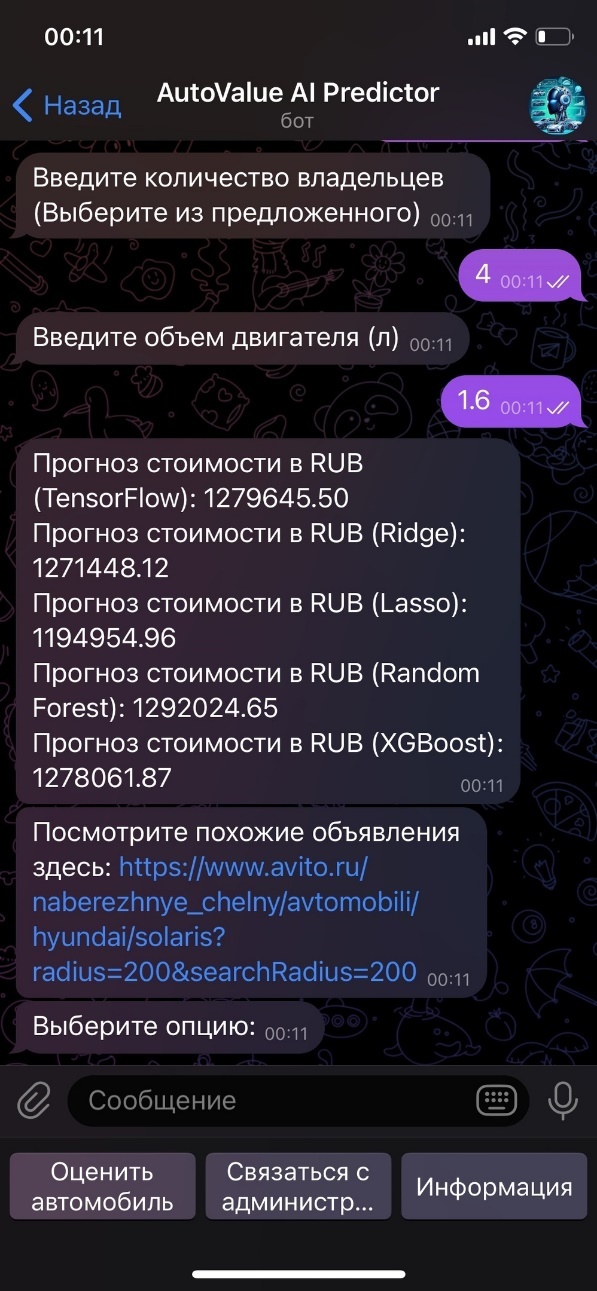


Рисунок 2.17 – Результаты прогнозирования и кнопки главного меню

* 1. **Тестирование разработанной программы**

Для подтверждения работы программы проведем ряд тестовых процедур. Для этого найдем реальные объявления о продаже автомобиля из разных ценовых категорий на авито, введем нужные характеристики в Telegram-бота и посмотрим на результаты.

**Тест 1.** Автомобиль Daewoo Nexia стоимостью 137000 рублей. Характеристики данного автомобиля можно увидеть на рисунке 2.18.

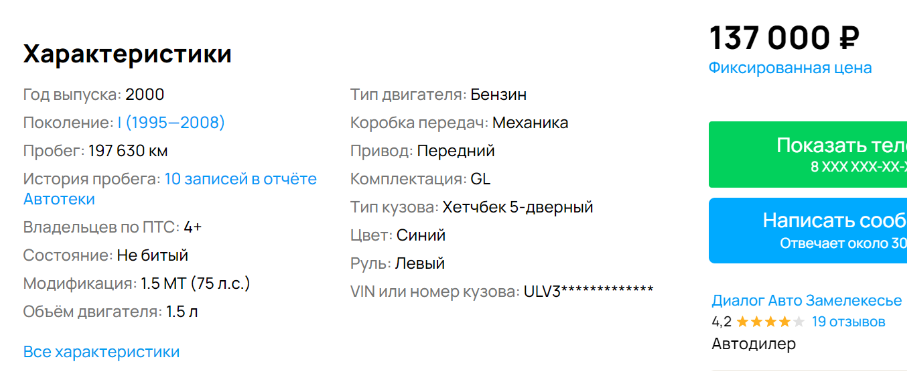


Рисунок 2.18 – Характеристики автомобиля Dawoo Nexia

Программа спрогнозировала следующие результаты (рисунок 2.19)

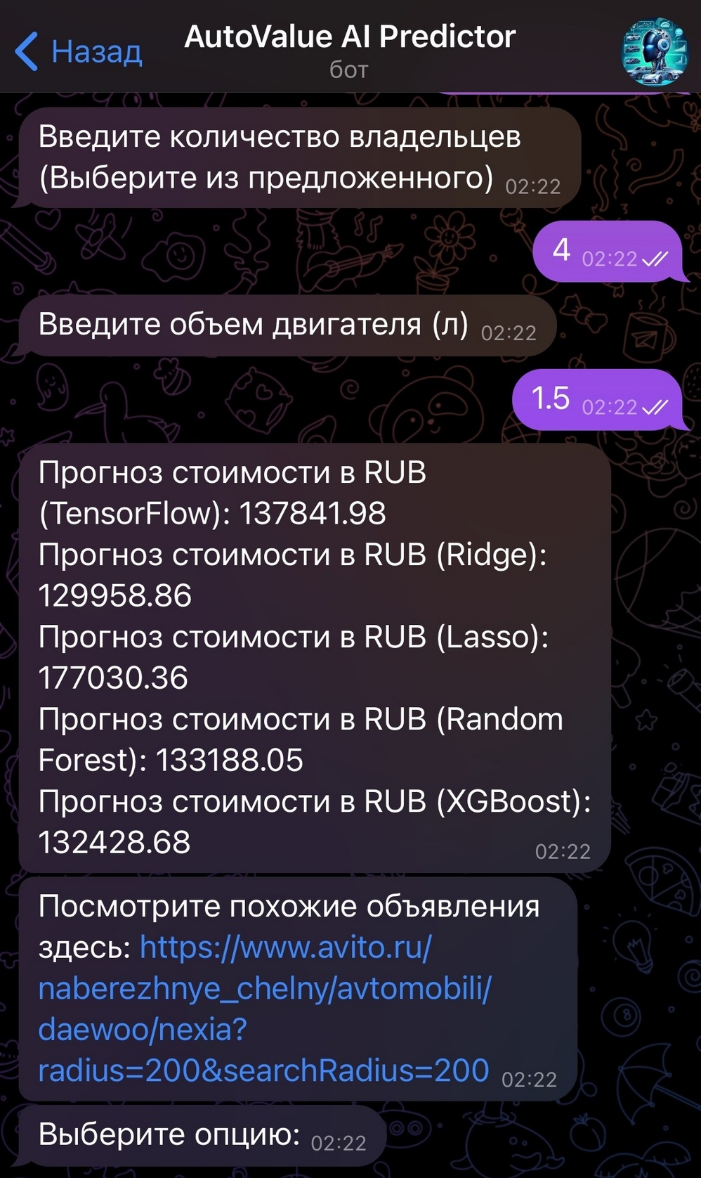


Рисунок 2.19 – Результаты прогнозов автомобиля Dawoo Nexia

В данном случае лучшую точность показала модель ИНС, с точностью почти до рубля, а худшую Lasso регрессия. Но в целом самая большая ошибка составила 40000 рублей, что является очень не плохим результатом.

**Тест 2.** Автомобиль ВАЗ (LADA) Granta стоимостью 590000 рублей. Характеристики данного автомобиля можно увидеть на рисунке 2.20, а спрогнозированные цены в результате работы программы на рисунке 2.21.

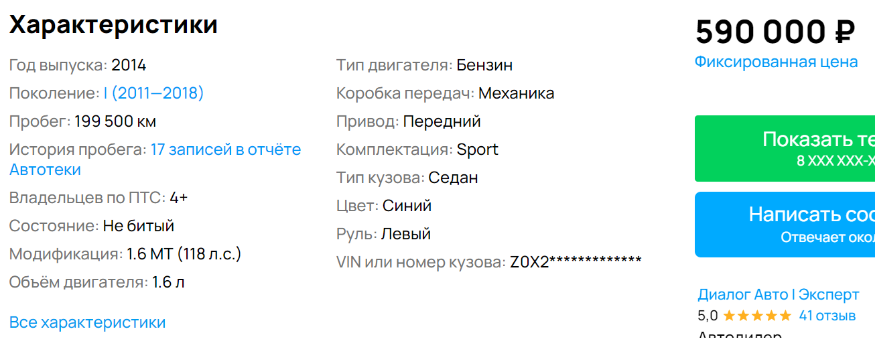


Рисунок 2.20 - Характеристики автомобиля LADA Granta

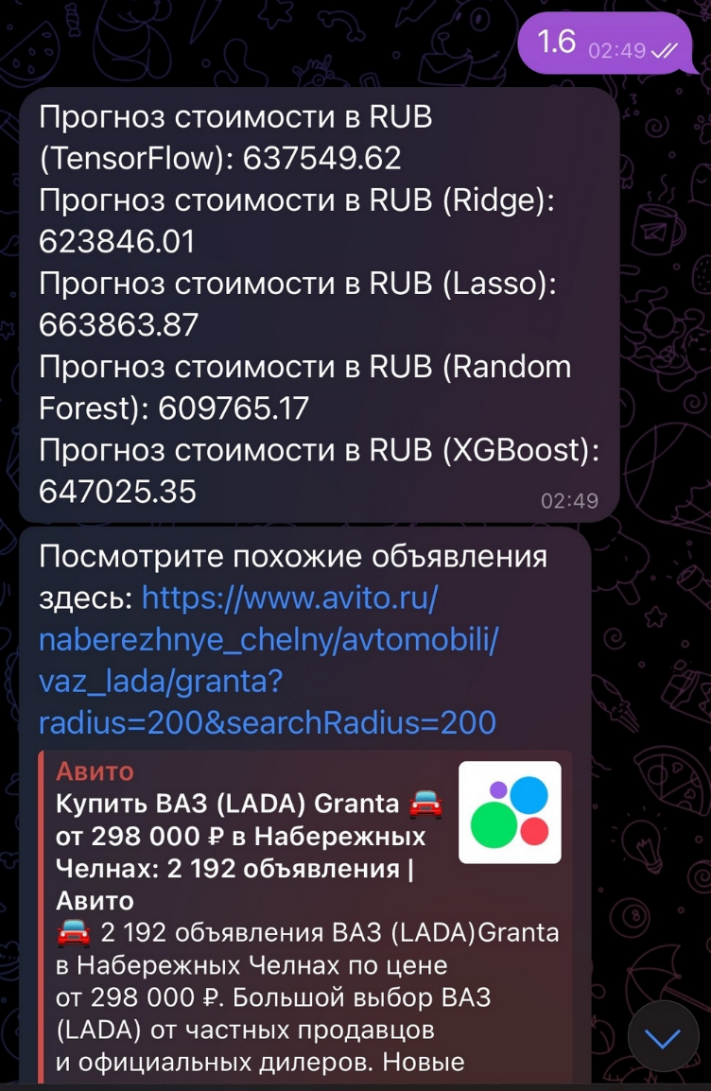


Рисунок 2.21 – Результаты прогнозов на автомобиль LADA Granta

Здесь лучшую точность показала модель Случайного леса, она ошиблась всего на 20000 рублей, а худшую на удивление снова Lasso регрессия. Ошибка у Lasso составила 63000 рублей.

**Тест 3.** Автомобиль Renault Sandero Stepway стоимостью на авито 880000 рублей. Характеристики данного автомобиля представлены на рисунке 2.22, а спрогнозированные цены в результате работы программы на рисунке 2.23.

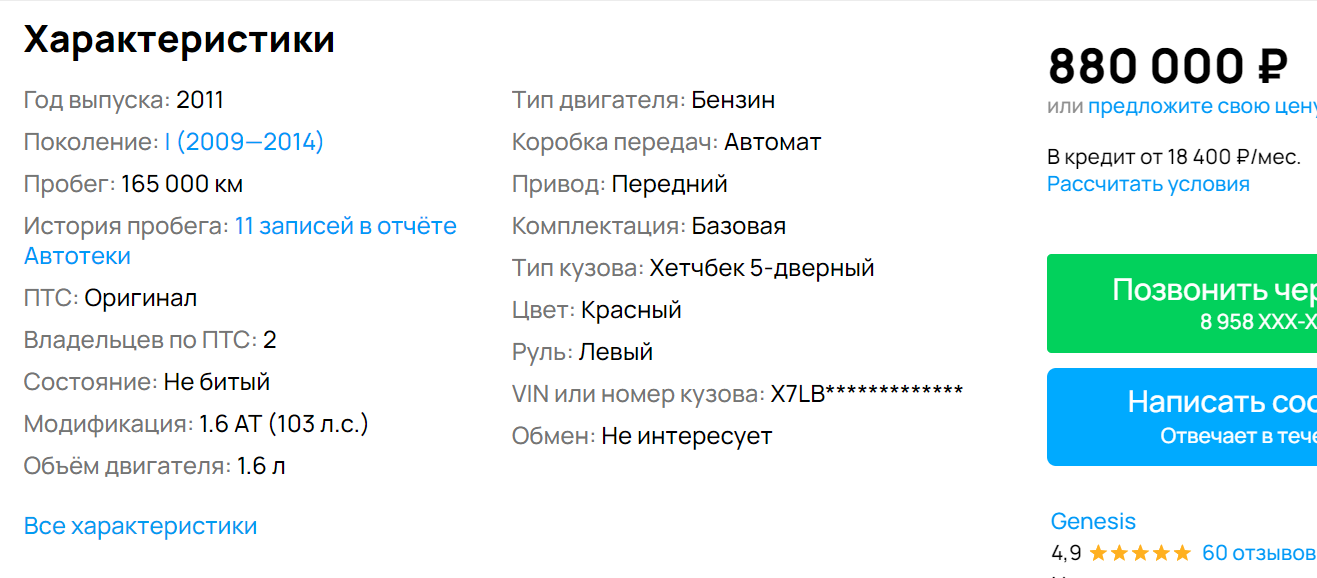


Рисунок 2.22 - Характеристики автомобиля Renault Sandero Stepway

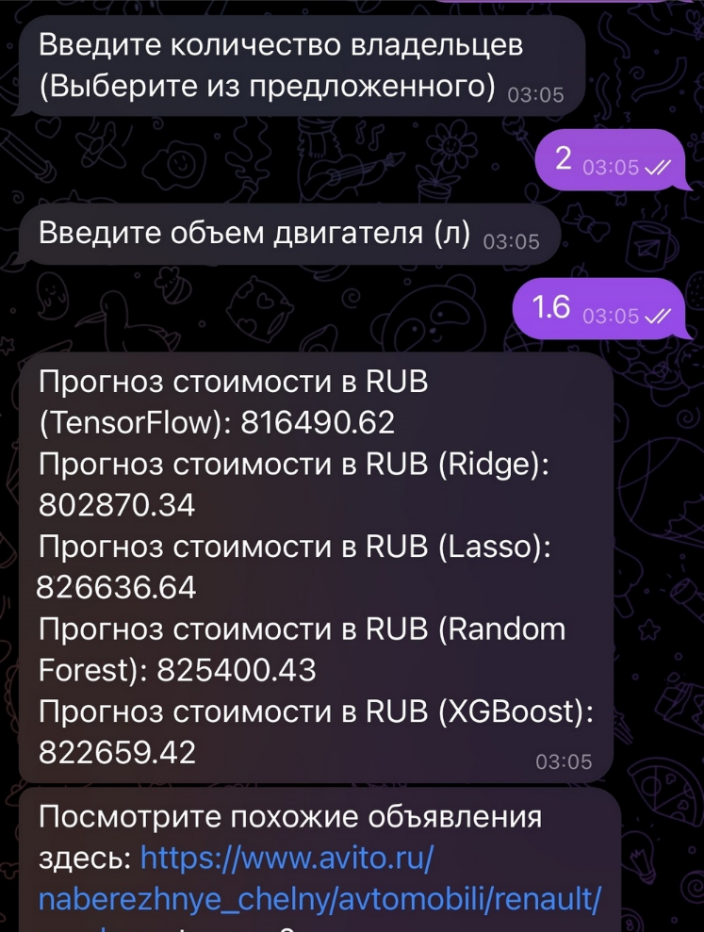


Рисунок 2.23 - Результаты прогнозов на автомобиль Renault Sandero Stepway

Лучшую точность показали модели Случайного леса и Lasso регрессия, они ошиблись на 55000 рублей примерно. Худший результат показала Линейная регрессия, она ошиблась на примерно 77000 рублей.

**Тест 4.** Автомобиль Kia Rio, цена продажи на авито 1398000 рублей. Характеристики данного автомобиля на рисунке 2.24, а спрогнозированные цены в результате работы программы на рисунке 2.25.

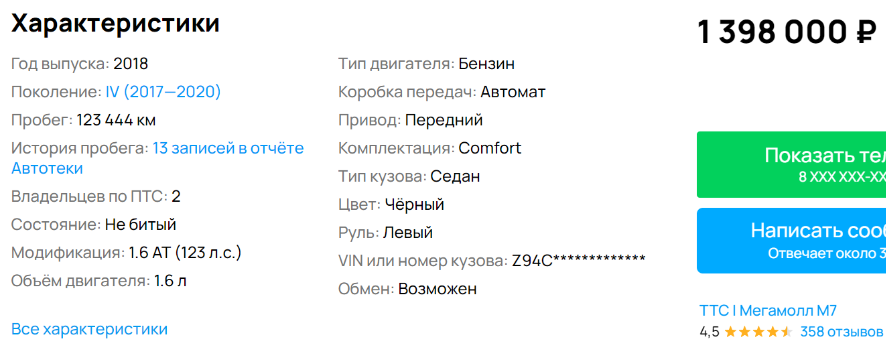


Рисунок 2.24 - Характеристики автомобиля Kia Rio

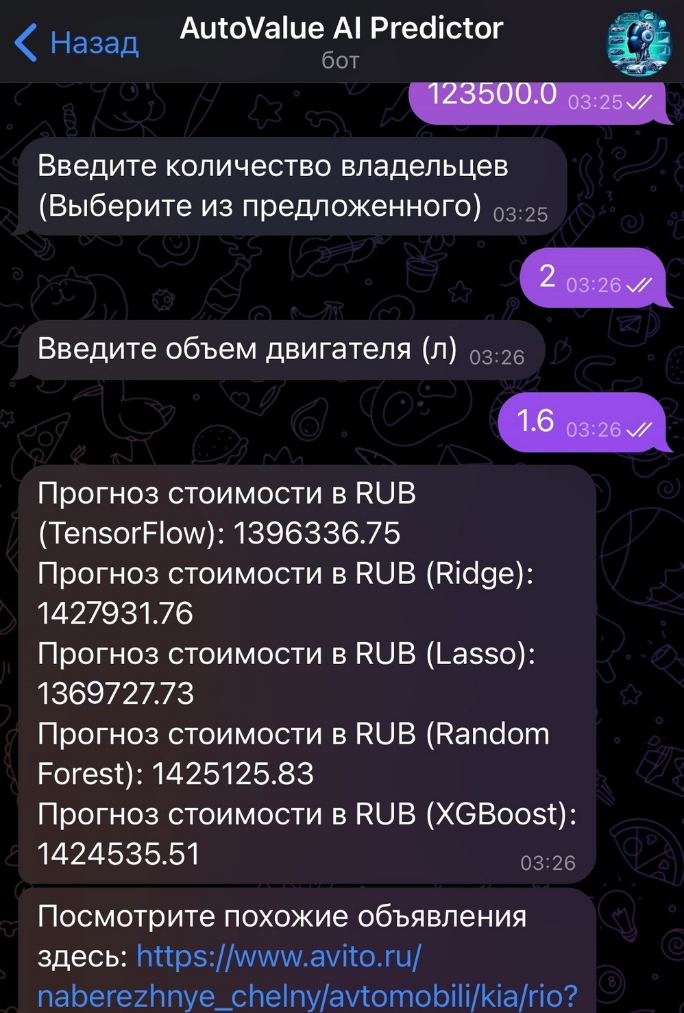


Рисунок 2.25 - Результаты прогнозов на автомобиль Kia Rio

Лучшую точность показала модель ИНС, она ошиблись всего на 2000 рублей. Самая плохая точность в данном случае у двух моделей: Линейная регрессия и Лассо регрессия. Их ошибка составила примерно 31000 рублей.

**Тест 5.** Автомобиль Volkswagen Touareg, цена продажи на сайте авито 2200000 рублей. Характеристики данного автомобиля представлены на рисунке 2.26, а спрогнозированные цены в результате работы программы на рисунке 2.27.

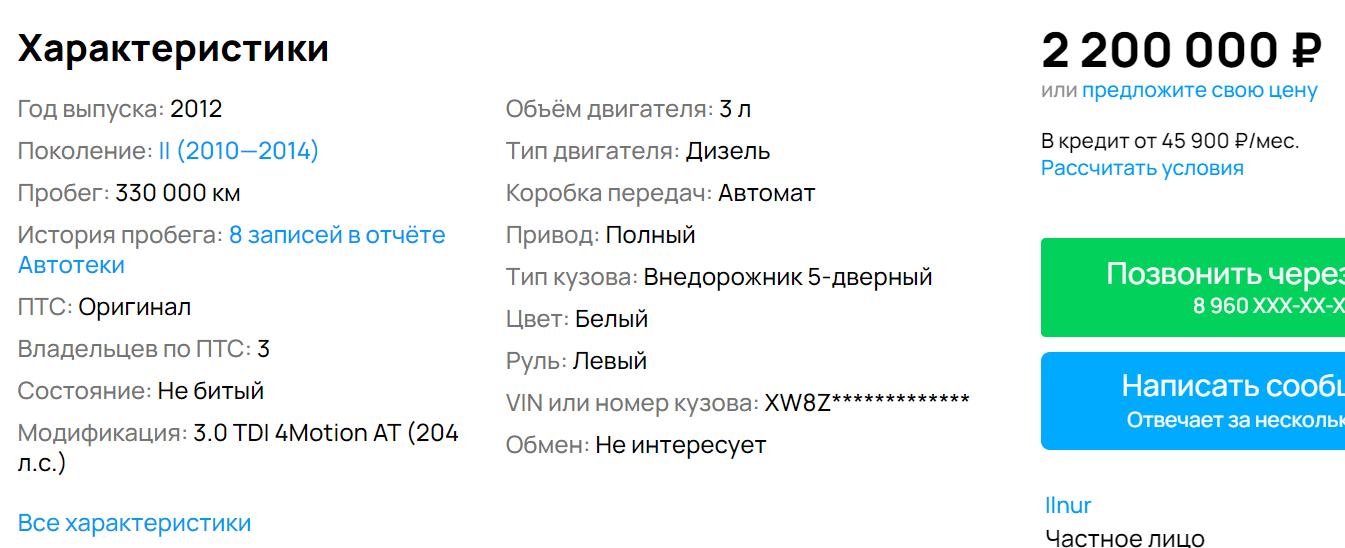


Рисунок 2.26 – Характеристики автомобиля Volkswagen Touareg

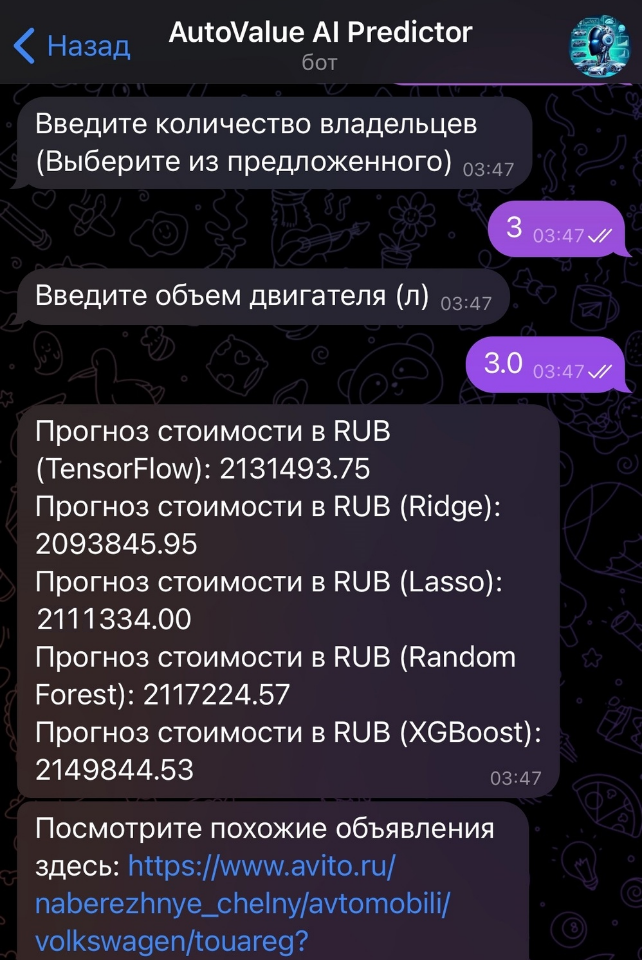


Рисунок 2.27 – Результаты прогнозов на автомобиль Volkswagen Touareg

Для данного автомобиля лучшую точность показала модель Экстремального градиентного бустинга, ее ошибка составила 50000 рублей. Самая плохая точность у Линейной регрессии, она ошиблась на 107000 рублей. Lasso ошиблась на 89000 рублей, Случайный лес на 83000 рублей, ИНС на 69000 рублей.

В общем и целом, все методы машинного обучения на реальных данных с авито показали себя очень хорошо, результаты ошибок совпали с результатами на тестовой выборке. Лучше всех прогнозировали цены модели Случайного леса и Искусственной нейронной сети. Хуже всех прогнозировали: Линейная регрессия и Лассо регрессия.

Дополнительно к тестированию моделей машинного обучения, важно уделить внимание тестированию графического интерфейса пользователя, который в данном случае реализован через Telegram-бота. Эффективность и удобство интерфейса напрямую влияют на восприятие приложения пользователями и их готовность его использовать.

Отклик и производительность.

В ходе тестирования бота особое внимание было уделено его способности быстро и корректно реагировать на ввод данных пользователем. Бот успешно справлялся с обработкой запросов в реальном времени, обеспечивая низкую задержку между сообщениями. Это особенно важно для поддержания интереса и вовлеченности пользователя, так как даже небольшие задержки могут негативно сказаться на пользовательском опыте.

Удобство использования.

Интерфейс Telegram-бота был разработан с учетом удобства использования:

1. Интуитивно понятные команды и подсказки помогают пользователям легко ориентироваться по функционалу бота.
2. Кнопки для быстрого ответа существенно упрощают процесс ввода данных, сокращая вероятность ошибок и ускоряя процесс взаимодействия.
3. Четкие инструкции и обратная связь на каждом шаге обеспечивают понимание текущего действия и следующих шагов, что улучшает общее взаимодействие с приложением.

Обработка ошибок.

Бот был оснащен механизмами обработки ошибок, что позволяло предотвратить его сбои и предоставлять пользователю полезные сообщения в случае некорректного ввода или других проблем. Это повышает доверие пользователей к стабильности и надежности приложения.

Время отклика.

При тестировании было отмечено, что время отклика бота на запросы пользователя остается стабильно низким даже при увеличении числа пользователей и запросов, что говорит о хорошей оптимизации и производительности бота.

# **ЗАКЛЮЧЕНИЕ**

В результате выполнения данной работы был проведен сравнительный анализ существующих методов машинного обучения. Затем, на основе полученных данных, были реализованы выбранные методы машинного обучения, включая разработку графического интерфейса в виде Telegram-бота для удобного взаимодействия с системой. Программный код был разработан на языке Python с использованием таких библиотек, как Tensorflow, Keras и Scikit-Learn. Для сбора данных о продаже подержанных автомобилей в Набережных Челнах и прилегающих регионах была создана специализированная программа. Собранные данные прошли этап первичной обработки, что позволило улучшить качество и точность прогнозирования цен на подержанные автомобили.

В ходе работы было применено пять различных методов машинного обучения. Тестирование показало, что модель, основанная на методе случайного леса, показала наилучшие результаты, в то время как линейная регрессия оказалась менее эффективной. Эти результаты подтверждают успешность выполнения поставленных задач и достижение основной цели работы — создания работоспособного приложения для прогнозирования стоимости подержанных автомобилей.

Результаты данного исследования могут быть применены автодилерами и частными лицами для более обоснованных решений при покупке автомобилей. Кроме того, разработанная система может служить основой для дальнейших исследований в области машинного обучения и его приложений в автомобильной индустрии.

Возможное дальнейшее развитие работы включает увеличение объёма и разнообразия данных для обучения, что потенциально приведёт к повышению точности прогнозов. Также стоит рассмотреть интеграцию дополнительных моделей машинного обучения и развитие пользовательского интерфейса для расширения функциональности и улучшения пользовательского опыта.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Бураков М. В. Нейронные сети и нейроконтроллеры : учебное пособие / М. В. Бураков. – Санкт-Петербург : ГУАП, 2013. – 284 с. - Текст: непосредственный.
2. Груздев А. В. Изучаем Pandas / А. В. Груздев, М. Хейдт. – Москва: "ДМК Пресс", 2019. – 700 с. – Текст: непосредственный.
3. Груздев А. В. Предварительная подготовка данных в Python. Том 1. Инструменты и валидация / А. В. Груздев, А. Раджараман, Ю. Лесковец. – Москва: ДМК Пресс, 2023. – 816 с. – Текст: непосредственный.
4. Гудфеллоу Я. Глубокое обучение / Я. Гудфеллоу, И. Бенджио, А. Курвилль; перевод с английского А. А. Слинкиной. – 2-е изд. испр. – Москва
5. : ДМК Пресс, 2018. – 652 с. - Текст : непосредственный.
6. Демиденко А. Telegram Bot. Руководство по созданию бота в мессенджере Телеграм / А. Демиденко – сетевое издание Leanpud, 2023. – 29 c. – Текст: непосредственный.
7. Жерон О. Прикладное машинное обучение с помощью Scikit- Learn и TensorFlow: концепции, инструменты и техники для создания интеллектуальных систем / О. Жерон ; перевод с английского Ю. Н. Артеменко. – Санкт-Петербург : ООО «Альфа-книга», 2018. – 688 с. - Текст: непосредственный.
8. Микелуччи У. Прикладное глубокое обучение. Подход к пониманию глубоких нейронных сетей на основе метода кейсов : перевод с английского / У. Микелуччи. – Санкт-Петербург : БХВ-Петербург, 2020. – 368 с. - Текст: непосредственный.
9. Миркин Б.Г. Введение в анализ данных. Учебник и практикум для бакалавриата и магистратуры / Б.Г. Миркин. – Москва : Юрайт, 2015. – 174 с. Текст: непосредственный.
10. Николенко С. Глубокое обучение / С. Николенко, А. Кадурин, Е. Архангельская. – Санкт-Петербург : Питер, 2018. – 480 с. - Текст: непосредственный.
11. Рафалович В. Data mining, или Интеллектуальный анализ данных для занятых. Практический курс / В. Рафалович. – Москва : СмартБук, 2014. – 110 с. Текст: непосредственный.
12. Хайкин С. Нейронные сети : полный курс / С. Хайкин. - 2-е изд. – Москва : Издательский дом «Вильямс», 2006. – 1104 с. - Текст: непосредственный.
13. Хошев П. Selenium Python / П. Хошев. – сетевое издание Leanpud, 2022. – 85 с. – Текст: непосредственный.
14. Шолле Ф. Глубокое обучение на Python / Ф. Шолле. – Санкт- Петербург : Питер, 2018. – 400 с. - Текст: непосредственный.
15. Шолле Ф. Глубокое обучение на R / Ф. Шолле. – Санкт- Петербург : Питер, 2018. – 400 с. - Текст: непосредственный.
16. Шумский С. А. Машинный интеллект. Очерки по теории машинного обучения и искусственного интеллекта / C. А. Шумский. – Москва : РИОР, 2019. – 340 с. - Текст: непосредственный.
17. Апарнев А. Н. Функции потерь в задаче обучения нейронной сети. / Апарнев А. Н. – Текст : электронный // 100byte.ru:[сайт] – 2016. – 13 мая – URL: [http://100byte.ru/stdntswrks/loss/loss.html](http://100byte.ru/stdntswrks/loss/loss.html%20) (дата обращения: 29.05.2024).
18. Алексеев Григорий Введение в Python. / Григорий Алексеев. – Текст: электронный // habr.com:[сайт]. – 2019. – 4 мая. – URL: <https://habr.com/ru/post/450474/> (дата обращения: 4.04.2024).
19. Шмиг Андрей Введение в машинное обучение. / Андрей Шмиг. – Текст: электронный // habr.com:[сайт]. – 2019. – 27 мая. – URL: [https://habr.com/ru/post/453558/](https://habr.com/ru/post/453558/%20) (дата обращения: 4.04.2024).
20. Радченко Виталий Открытый курс машинного обучения. Тема 5. Композиции: бэггинг, случайный лес. / Виталий Радченко. – Текст: электронный // habr.com:[сайт]. – 2017. – 27 мар. – URL: [https://habr.com/ru/company/ods/blog/324402/](https://habr.com/ru/company/ods/blog/324402/%20) (дата обращения: 1.04.2024).
21. Павленко Дмитрий Введение в машинное обучение и искусственные нейронные сети / Дмитрий Павленко. – Текст: электронный // github.io:[сайт]. – 2020 – 22 января. – URL: [https://foobar167.github.io/page/vvedeniye-v-mashinnoye-obucheniye-i-iskusstvennyye-neyronnyye-seti.html#ref2.5](https://foobar167.github.io/page/vvedeniye-v-mashinnoye-obucheniye-i-iskusstvennyye-neyronnyye-seti.html%23ref2.5) (дата обращения: 02.04.2024).
22. Нестеров Павел Базовые принципы машинного обучения на примере линейной регрессии. / Павел Нестеров. - Текст : электронный // habr.com:[сайт]. – 2017. – 20 фев. – URL: [https://habr.com/ru/company/ods/blog/322076/](https://habr.com/ru/company/ods/blog/322076/%20) (дата обращения: 2.04.2024).
23. Старовойтов В. В. Нормализация данных в машинном обучении / В. В. Старовойтов, Ю. И. Голуб – Текст : электронный // inf.grid.by:[сайт]. – 2021 – 18 марта. 0 URL:[https://doi.org/10.37661/1816-0301-2021-18-3-83-96](https://doi.org/10.37661/1816-0301-2021-18-3-83-96%20) (дата обращения: 16.04.2024)
24. Kashnitsky Yury Открытый курс машинного обучения. Тема 4. Линейные модели классификации и регрессии. / Yury Kashnitsky. – Текст : электронный // habr.com:[сайт]. – 2017. – 20 мар. – URL: [https://habr.com/ru/company/ods/blog/323890/](https://habr.com/ru/company/ods/blog/323890/%20) (дата обращения: 4.04.2024).
25. Kashnitsky Yury Открытый курс машинного обучения. Тема 3. Классификация, деревья решений и метод ближайших соседей. / Yury Kashnitsky. – Текст : электронный // habr.com:[сайт]. – 2017. – 13 мар. – URL: [https://habr.com/ru/company/ods/blog/322534/](https://habr.com/ru/company/ods/blog/322534/%20) (дата обращения: 4.04.2024).

# **ПРИЛОЖЕНИЕ**

## **Приложение A. Листинг кода парсера**

from selenium import webdriver  
from selenium.common import NoSuchElementException  
from selenium\_stealth import stealth  
import time  
from selenium.webdriver.chrome.service import Service  
from selenium.webdriver.common.by import By  
import csv  
import os  
import random  
#создаем csv  
desktop\_path = os.path.expanduser("~\\OneDrive\\Рабочий стол")  
csv\_file\_path = os.path.join(desktop\_path, "avito\_data16\_mersedes.csv")  
csv\_headers = ["Marka", "Model", "Год выпуска", "Поколение", "Пробег", "История пробега", "ПТС", "Владельцев по ПТС", "Состояние", "Модификация", "Объём двигателя", "Тип двигателя", "Коробка передач", "Привод", "Комплектация", "Тип кузова", "Цвет", "Руль", "VIN или номер кузова", "Обмен"]  
with open(csv\_file\_path, mode='w', newline='') as csv\_file:  
 writer = csv.writer(csv\_file)  
 writer.writerow(csv\_headers)

# обход защиты  
 service = webdriver.ChromeOptions()  
 service.add\_argument("start-maximized")  
 service.add\_experimental\_option("excludeSwitches", ["enable-automation"])  
 service.add\_experimental\_option('useAutomationExtension', False)  
  
 service = Service('C:\\Users\\пользватель\\PycharmProjects\\pythonProject\\drivers\\chromedriver.exe')  
 driver = webdriver.Chrome(service=service)  
  
 stealth(driver,  
 languages=["ru-RU", "ru"],  
 vendor="Google Inc.",  
 platform="Win64",  
 webgl\_vendor="Intel Inc.",  
 renderer="Intel Iris OpenGL Engine",  
 fix\_hairline=True,  
 )  
 data = set()  
 # парсим ссылки  
 for page in range(1, 14):  
 url = f"https://www.avito.ru/naberezhnye\_chelny/avtomobili/mercedes-benz/levyy\_rul-ASgBAgICAkTwCqyKAeC2DeiYKA?cd=1&f=ASgBAgICBUTwCqyKAfIKsIoBhhTI5gHgtg3omCj68A\_ou\_cC&p={page}&radius=200&searchRadius=200"  
 driver.get(url)  
 time.sleep(10+random.random())  
 blocks = driver.find\_element(By.CSS\_SELECTOR, "#app > div > div.styles-singlePageWrapper-eKDyt > div > div.index-center-\_TsYY.index-center\_withTitle-\_S7ge.index-center\_noMarginTop-xAh5X.index-centerWide-\_7ZZ\_.index-center\_marginTop\_1-ewXHO > div.index-inner-dqBR5.index-innerCatalog-ujLwf > div.index-content-\_KxNP > div.index-root-KVurS > div.items-items-kAJAg")  
 posts = blocks.find\_elements(By.CLASS\_NAME, "styles-module-theme-CRreZ")  
 for post in posts:  
 try:  
 title = post.find\_element(By.CLASS\_NAME, "iva-item-slider-pYwHo").find\_element(By.TAG\_NAME, "a").get\_attribute( "href")  
 data.add(title)  
 except NoSuchElementException:  
 continue  
 data\_list = list(data)  
#Собираем нужные данные  
 for url in data\_list:  
 driver.get(url)  
 time.sleep(17+random.random()+random.random()+random.random())  
 try:  
 marka = driver.find\_element(By.CSS\_SELECTOR,  
 "#app > div > div.index-root-k1Ib4.index-responsive-aOpFS.index-page\_default-\_b5bD > div:nth-child(1) > div > div.style-item-view-PCYlM > div.style-item-navigation-In5Jr > div:nth-child(2) > span:nth-child(5) > a > span").text  
 model = driver.find\_element(By.CSS\_SELECTOR,  
 "#app > div > div.index-root-k1Ib4.index-responsive-aOpFS.index-page\_default-\_b5bD > div:nth-child(1) > div > div.style-item-view-PCYlM > div.style-item-navigation-In5Jr > div:nth-child(2) > span:nth-child(6) > a > span").text  
 params\_elements = driver.find\_elements(By.CLASS\_NAME, 'params-paramsList\_\_item-appQw')  
 price = driver.find\_element(By.CSS\_SELECTOR,  
 "#app > div > div.index-root-k1Ib4.index-responsive-aOpFS.index-page\_default-\_b5bD > div:nth-child(1) > div > div.style-item-view-PCYlM > div.style-item-view-content-SDgKX > div.style-item-view-content-right-rxJqW > div.style-item-view-info-HCcXB > div > div > div.style-item-view-price-block-WSyYk > div > div.styles-module-theme-CRreZ > div > div:nth-child(1) > div > span > span > span:nth-child(1)").text  
 params\_data = {  
 'Год выпуска': None,  
 'Поколение': None,  
 'Пробег': None,  
 'История пробега': None,  
 'ПТС': None,  
 'Владельцев по ПТС': None,  
 'Состояние': None,  
 'Модификация': None,  
 'Объём двигателя': None,  
 'Тип двигателя': None,  
 'Коробка передач': None,  
 'Привод': None,  
 'Комплектация': None,  
 'Тип кузова': None,  
 'Цвет': None,  
 'Руль': None,  
 'VIN или номер кузова': None,  
 'Обмен': None  
 }  
 for param\_element in params\_elements:  
 param\_parts = param\_element.text.split(':')  
 if len(param\_parts) == 2:  
 param\_name = param\_parts[0].strip()  
 param\_value = param\_parts[1].strip()  
 if param\_name in params\_data:  
 params\_data[param\_name] = param\_value  
 params\_values = list(params\_data.values())  
 writer.writerow([marka, model] + params\_values + [price])  
 except NoSuchElementException:  
 continue  
 driver.quit()

## **Приложение Б. Листинг кода первичной обработки**

import pandas as pd  
import numpy as np  
import pandas as pd  
pd.set\_option('display.max\_columns', None)  
import warnings  
warnings.filterwarnings('ignore')  
cars = pd.read\_csv("C:/Users/пользватель/OneDrive/Рабочий стол/eto\_ono.csv")  
cars = cars[~cars['model'].isna()]  
cars = cars.dropna(axis=1, how='all')  
cars.duplicated().value\_counts()  
# Заменим пропущенные значения средним значением  
cars['horse\_power'] = cars['horse\_power'].fillna(int(cars['horse\_power'].mean()))  
# Заменяем пропущенные значения модой  
cars['configuration'] = cars['configuration'].fillna(cars['configuration'].mode()[0])  
cars['complectation'] = cars['complectation'].fillna(cars['complectation'].mode()[0])  
cars['body\_type'] = cars['body\_type'].fillna(cars['body\_type'].mode()[0])  
cars['drive\_type'] = cars['drive\_type'].fillna(cars['drive\_type'].mode()[0])  
cars['engine\_type'] = cars['engine\_type'].fillna(cars['engine\_type'].mode()[0])  
cars['transmission'] = cars['transmission'].fillna(cars['transmission'].mode()[0])  
# cars['price\_rub'] = cars['price\_rub'].astype(str).apply(lambda s: ''.join([x for x in s if x.isdigit()])).astype('int32')  
print("Cars with restyling:", sum(cars['generation'].apply(lambda x: 'рестайлинг' in x)))  
cars['restyling'] = cars['generation'].apply(lambda x: 'рестайлинг' in x).map({True: 'Да', False: 'Нет'})  
cars['generation'] = cars['generation'].apply(lambda x: x.replace('рестайлинг ', ''))  
cars['generation'] = cars['generation'].apply(lambda x: x.split()[0])  
cars['engine\_volume'] = cars['configuration'].str.extract(r'(\d+\.\d+)').astype(float)  
packages = {  
 'Базовая': 'Base', 'Base': 'Base',  
 'SE': 'SE', 'Особая серия': 'SE',  
 'Lux': 'Luxe', 'Люкс': 'Luxe',  
 'Норма': 'Norma', 'Norma': 'Norma',  
 'Sport': 'Sport', 'HSE': 'HSE',  
 'Стандарт': 'Standard', 'Standart': 'Standard',  
 'Комфорт': 'Comfort', 'Confort': 'Comfort',  
 'Comfort': 'Comfort', 'Premium': 'Premium',  
 'Enjoy': 'Enjoy', 'Executive': 'Executive',  
 'Special edition': 'SE', 'Limited Edition': 'LE',  
 'Limited': 'LE', 'Active': 'Active',  
 'Prestige': 'Prestige', 'Invite': 'Invite',  
 'Business': 'Business', 'Trendline': 'Trend'  
}  
def same\_package(value):  
 for package in packages:  
 if package in value:  
 return packages.get(package)  
 return value  
cars['complectation'] = cars['complectation'].apply(lambda x: same\_package(x))  
cars = cars[(cars['price\_rub'] < cars.price\_rub.quantile(0.99)) & (cars['price\_rub'] > cars.price\_rub.quantile(0.01))]  
cars = cars[~((cars['km\_age'] <= 10000) | (cars['km\_age'] >= 900000))]  
cars = cars.drop('configuration', axis=1)  
cars.info()  
cars.to\_csv('C:/Users/пользватель/OneDrive/Рабочий стол/eto\_ono2.csv', index=False, encoding='cp1251')

## **Приложение В. Листинг кода машинного обучения**

import numpy as np  
import pandas as pd

from sklearn.preprocessing import LabelBinarizer, StandardScaler, PolynomialFeatures  
from sklearn.metrics import mean\_squared\_error, r2\_score, mean\_absolute\_error  
from sklearn.model\_selection import train\_test\_split, GridSearchCV, KFold  
from sklearn.linear\_model import Ridge, SGDRegressor, Lasso, ElasticNet  
from sklearn.ensemble import RandomForestRegressor  
import xgboost as xgb  
import warnings

warnings.filterwarnings('ignore')  
RANDOM\_STATE = 80  
# Load data  
cars = pd.read\_csv('C:/Users/пользватель/OneDrive/Рабочий стол/eto\_ono6.csv')  
# Creating max\_price\_for\_category column  
max\_prices = cars.groupby('price\_category')['price\_rub'].max().rename('max\_price\_for\_category')  
cars = cars.join(max\_prices, on='price\_category')  
# Shuffle data  
cars = cars.sample(frac=1, random\_state=RANDOM\_STATE)  
CATEGORICAL\_FEATURES = ['mark', 'model', 'generation', 'body\_type', 'engine\_type',  
 'transmission', 'color', 'drive\_type', 'wheel', 'complectation', 'price\_category']  
NUMERIC\_FEATURES = ['horse\_power', 'year', 'km\_age', 'owners\_count', 'engine\_volume']  
def preprocess\_features(dataframe):  
 lb = LabelBinarizer()  
 dataframe['condition'] = lb.fit\_transform(dataframe['condition'])  
 dataframe['restyling'] = lb.fit\_transform(dataframe['restyling'])  
 dataframe = pd.get\_dummies(dataframe, columns=CATEGORICAL\_FEATURES)  
 print(f"We have {dataframe.shape[1]} features after one-hot encoding")  
 scaler = StandardScaler()  
 poly = PolynomialFeatures(interaction\_only=True)  
 numeric\_data = scaler.fit\_transform(dataframe[NUMERIC\_FEATURES])  
 numeric\_data\_poly = poly.fit\_transform(numeric\_data)  
 poly\_features\_names = [f"poly\_{i}" for i in range(numeric\_data\_poly.shape[1])]  
 dataframe = dataframe.drop(NUMERIC\_FEATURES, axis=1)  
 dataframe[poly\_features\_names] = numeric\_data\_poly  
 return dataframe  
def make\_prediction(model, X\_train, X\_test, y\_train, y\_test, params\_fixed, params\_grid=None):  
 if params\_grid:  
 estimator = GridSearchCV(model(\*\*params\_fixed), params\_grid, cv=KFold(n\_splits=3), verbose=True)  
 else:  
 estimator = model(\*\*params\_fixed)  
 estimator.fit(X\_train, y\_train)  
 y\_preds\_train = estimator.predict(X\_train)  
 y\_preds\_test = estimator.predict(X\_test)  
 # Denormalize predictions  
 y\_preds\_train\_unnormalized = np.exp(y\_preds\_train) \* X\_train['max\_price\_for\_category']  
 y\_preds\_test\_unnormalized = np.exp(y\_preds\_test) \* X\_test['max\_price\_for\_category']  
 y\_train\_unnormalized = np.exp(y\_train) \* X\_train['max\_price\_for\_category']  
 y\_test\_unnormalized = np.exp(y\_test) \* X\_test['max\_price\_for\_category']  
  
 # Recalculate metrics for denormalized values  
 mse\_train = mean\_squared\_error(y\_train\_unnormalized, y\_preds\_train\_unnormalized)  
 r2\_train = r2\_score(y\_train\_unnormalized, y\_preds\_train\_unnormalized)  
 mae\_train = mean\_absolute\_error(y\_train\_unnormalized, y\_preds\_train\_unnormalized)  
 mse\_test = mean\_squared\_error(y\_test\_unnormalized, y\_preds\_test\_unnormalized)  
 r2\_test = r2\_score(y\_test\_unnormalized, y\_preds\_test\_unnormalized)  
 mae\_test = mean\_absolute\_error(y\_test\_unnormalized, y\_preds\_test\_unnormalized)  
 print(f'Train MSE: {mse\_train}, Train R2: {r2\_train}, Train MAE: {mae\_train}')  
 print(f'Test MSE: {mse\_test}, Test R2: {r2\_test}, Test MAE: {mae\_test}')  
 if params\_grid:  
 print("Best parameters:", estimator.best\_params\_)  
 return estimator, y\_train\_unnormalized, y\_preds\_train\_unnormalized, y\_test\_unnormalized, y\_preds\_test\_unnormalized  
# Preprocessing data  
cars = preprocess\_features(cars)  
# Prepare data for training  
y = np.log(cars['normalized\_price']) # Log-normalizing price  
X = cars.drop(['price\_rub', 'normalized\_price'], axis=1)  
# Splitting data into training and testing  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=RANDOM\_STATE, shuffle=True)  
# Train Lasso model with Grid Search  
lasso\_params\_fixed = {'random\_state': RANDOM\_STATE}  
lasso\_params\_grid = {'alpha': np.linspace(1e-05, 1e-03, num=10)}  
print("\tLasso model with Grid Search\n")  
lasso\_model, lasso\_y\_train, lasso\_y\_preds\_train, lasso\_y\_test, lasso\_y\_preds\_test = make\_prediction(Lasso, X\_train, X\_test, y\_train, y\_test, lasso\_params\_fixed, lasso\_params\_grid)  
  
# Train Ridge model  
print("\tRidge model (Baseline)\n")  
ridge\_params = {'random\_state': RANDOM\_STATE}  
ridge\_model, ridge\_y\_train, ridge\_y\_preds\_train, ridge\_y\_test, ridge\_y\_preds\_test = make\_prediction(Ridge, X\_train, X\_test, y\_train, y\_test, ridge\_params)  
# Train Random Forest  
print("\tRandom Forest (Baseline)\n")  
rf\_params = {'random\_state': RANDOM\_STATE}  
rf\_model, rf\_y\_train, rf\_y\_preds\_train, rf\_y\_test, rf\_y\_preds\_test = make\_prediction(RandomForestRegressor, X\_train, X\_test, y\_train, y\_test, rf\_params)  
# Train XGBoost model  
print("\tXGBoost model (Baseline)\n")  
xgb\_params\_fixed = {'random\_state': RANDOM\_STATE, 'use\_label\_encoder': False, 'eval\_metric': 'rmse'}  
xgb\_params\_grid = {  
 'n\_estimators': [100, 200],  
 'max\_depth': [3, 5, 7],  
 'learning\_rate': [0.01, 0.1, 0.2]  
}  
xgb\_model, xgb\_y\_train, xgb\_y\_preds\_train, xgb\_y\_test, xgb\_y\_preds\_test = make\_prediction(xgb.XGBRegressor, X\_train, X\_test, y\_train, y\_test, xgb\_params\_fixed, xgb\_params\_grid)

## **Приложение Г. Листинг кода графического интерфейса**

import telebot  
from telebot import types  
from tensorflow.keras.models import load\_model  
import joblib  
import numpy as np  
import pandas as pd  
bot\_token = '6763430520:AAHHf5ueF6YI70FRvggUBp-ZGmizvjW3Yko'  
bot = telebot.TeleBot(bot\_token)  
# Загрузка моделей и объектов предварительной обработки  
category\_model = load\_model('path\_to\_my\_model/category\_model.h5')  
price\_model = load\_model('path\_to\_my\_model/price\_model.h5')  
ridge\_model = joblib.load('path\_to\_my\_model/ridge\_model.joblib')  
lasso\_model = joblib.load('path\_to\_my\_model/lasso\_model.joblib')  
rf\_model = joblib.load('path\_to\_my\_model/rf\_model.joblib')  
xgb\_model = joblib.load('path\_to\_my\_model/xgb\_model.joblib')  
onehot\_encoder = joblib.load('path\_to\_my\_model/onehot\_encoder.joblib')  
scaler = joblib.load('path\_to\_my\_model/scaler.joblib')  
max\_price\_dict = pd.read\_csv('path\_to\_my\_model/max\_price\_for\_category.csv').set\_index('price\_category')['max\_price\_for\_category'].to\_dict()  
FEATURES = [  
 ('mark', "Введите марку автомобиля"),  
 ('model', "Введите модель автомобиля"),  
 ('generation', "Введите поколение автомобиля"),  
 ('body\_type', "Введите тип кузова"),  
 ('engine\_type', "Введите тип двигателя (Выберите из предложенного)"),  
 ('transmission', "Введите тип трансмиссии (Выберите из предложенного)"),  
 ('color', "Введите цвет автомобиля"),  
 ('drive\_type', "Введите тип привода (Выберите из предложенного)"),  
 ('wheel', "Введите расположение руля (Выберите из предложенного)"),  
 ('complectation', "Введите комплектацию"),  
 ('restyling', "Автомобиль рестайлинговый? (Да/Нет)"),  
 ('condition', "Введите состояние автомобиля (Выберите из предложенного)"),  
 ('horse\_power', "Введите мощность двигателя (л.с.)"),  
 ('year', "Введите год выпуска"),  
 ('km\_age', "Введите пробег (км)"),  
 ('owners\_count', "Введите количество владельцев (Выберите из предложенного)"),  
 ('engine\_volume', "Введите объем двигателя (л)")  
]  
user\_data = {}  
def preprocess\_input(user\_df, onehot\_encoder, scaler):  
 user\_df\_categorical = onehot\_encoder.transform(user\_df[[f[0] for f in FEATURES[:12]]]).toarray()  
 user\_df\_numeric = scaler.transform(user\_df[[f[0] for f in FEATURES[12:]]])  
 return np.concatenate([user\_df\_categorical, user\_df\_numeric], axis=1)  
@bot.message\_handler(commands=['start'])  
def send\_welcome(message):  
 welcome\_text = (  
 "Добро пожаловать в AutoValue AI Predictor, ваш интеллектуальный помощник на рынке подержанных автомобилей! "  
 "Используя передовые алгоритмы машинного обучения, наш бот предоставляет вам точные оценки цен на подержанные "  
 "автомобили в режиме реального времени. Распрощайтесь с догадками и используйте возможности больших данных "  
 "для принятия обоснованных решений, покупаете ли вы, продаете или просто просматриваете веб-страницы. "  
 "Просто введите данные о вашем автомобиле, и пусть AutoValue AI Predictor за считанные секунды определит его "  
 "истинную рыночную стоимость."  
 )  
 markup = types.ReplyKeyboardMarkup(resize\_keyboard=True)  
 btn1 = types.KeyboardButton("Оценить автомобиль")  
 btn2 = types.KeyboardButton("Связаться с администратором")  
 btn3 = types.KeyboardButton("Информация")  
 markup.add(btn1, btn2, btn3)  
 bot.send\_message(message.chat.id, welcome\_text, reply\_markup=markup)  
def send\_main\_menu(chat\_id):  
 welcome\_text = (  
 "Выберите опцию:"  
 )  
 markup = types.ReplyKeyboardMarkup(resize\_keyboard=True)  
 btn1 = types.KeyboardButton("Оценить автомобиль")  
 btn2 = types.KeyboardButton("Связаться с администратором")  
 btn3 = types.KeyboardButton("Информация")  
 markup.add(btn1, btn2, btn3)  
 bot.send\_message(chat\_id, welcome\_text, reply\_markup=markup)  
@bot.message\_handler(func=lambda message: message.text == "Связаться с администратором")  
def contact\_admin(message):  
 bot.send\_message(message.chat.id, "Если возникли вопросы или предложения о сотрудничестве, пожалуйста, напишите @ioannserg")  
@bot.message\_handler(func=lambda message: message.text == "Информация")  
def show\_info(message):  
 bot.send\_message(message.chat.id, "Данный бот не дает гарантию на точное прогнозированние цены. "  
 "Средняя ошибка прогноза составляет ≈ 40-60 тысяч рублей. "  
 "Бот разработан в целях написания и защиты дипломной работы. "  
 "Если бот вышел из строя, введите /start "  
 "По всем вопросам обращаться к создателю @ioannserg")  
  
@bot.message\_handler(func=lambda message: message.text == "Оценить автомобиль")  
def evaluate\_car(message):  
 ask\_feature(message, 0)  
  
def ask\_feature(message, index):  
 feature\_name, feature\_prompt = FEATURES[index]  
 markup = types.ReplyKeyboardMarkup(resize\_keyboard=True, one\_time\_keyboard=True)  
 if feature\_name == 'engine\_type':  
 markup.add('Бензин', 'Дизель', 'Электро')  
 elif feature\_name == 'transmission':  
 markup.add('Автомат', 'Механика', 'Робот', 'Вариатор')  
 elif feature\_name == 'drive\_type':  
 markup.add('Передний', 'Задний', 'Полный')  
 elif feature\_name == 'wheel':  
 markup.add('Левый', 'Правый')  
 elif feature\_name == 'restyling':  
 markup.add('Да', 'Нет')  
 elif feature\_name == 'condition':  
 markup.add('отличное', 'требует ремонта')  
 elif feature\_name == 'owners\_count':  
 markup.add('1', '2', '3', '4')  
 msg = bot.send\_message(message.chat.id, feature\_prompt, reply\_markup=markup if markup.keyboard else None)  
 bot.register\_next\_step\_handler(msg, process\_feature, index)  
def process\_feature(message, index):  
 user\_data[FEATURES[index][0]] = message.text  
 next\_index = index + 1  
 if next\_index < len(FEATURES):  
 ask\_feature(message, next\_index)  
 else:  
 predict\_price(message.chat.id)  
def predict\_price(chat\_id):  
 try:  
 # Создание DataFrame из собранных данных пользователя  
 user\_df = pd.DataFrame([user\_data])  
  
 # Предварительная обработка введенных данных  
 processed\_input = preprocess\_input(user\_df, onehot\_encoder, scaler)  
  
 # Получение категории цены с помощью модели классификации  
 predicted\_category = category\_model.predict(processed\_input)  
 predicted\_category = np.argmax(predicted\_category, axis=1)[0]  
  
 # Установка категории цены в DataFrame  
 user\_df['price\_category'] = predicted\_category  
 # Вычисление максимального коэффициента цены для данной категории  
 max\_price\_factor = max\_price\_dict.get(predicted\_category, 1)  
 # Получение предсказаний от различных моделей  
 predictions = {  
 "TensorFlow": price\_model.predict(processed\_input),  
 "Ridge": ridge\_model.predict(processed\_input),  
 "Lasso": lasso\_model.predict(processed\_input),  
 "Random Forest": rf\_model.predict(processed\_input),  
 "XGBoost": xgb\_model.predict(processed\_input)  
 }  
 # Форматирование и отправка результатов предсказания  
 results = []  
 for name, pred in predictions.items():  
 price = np.exp(pred[0]) \* max\_price\_factor  
 if isinstance(price, np.ndarray):  
 price = price.item()  
 results.append(f"Прогноз стоимости в RUB ({name}): {price:.2f}")  
  
 bot.send\_message(chat\_id, "\n".join(results))  
 mark = user\_data['mark'].lower()  
 if mark == 'ваз (lada)':  
 mark = 'vaz\_lada'  
 else:  
 mark = mark  
 model = user\_data['model'].lower().replace(' ', '\_')  
 avito\_url = f"https://www.avito.ru/naberezhnye\_chelny/avtomobili/{mark}/{model}?radius=200&searchRadius=200"  
 bot.send\_message(chat\_id, f"Посмотрите похожие объявления здесь: {avito\_url}")  
 except Exception as e:  
 # Обработка любых исключений, возникших во время предсказания  
 bot.send\_message(chat\_id,  
 f"Произошла ошибка при расчете стоимости: {e}\nНе правильно введенные данные или сбой в системе.")  
 finally:  
 # Возвращение пользователя в главное меню  
 send\_main\_menu(chat\_id)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 bot.polling()