ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»
**Московский институт электроники и математики**

# Решение задачи регрессии с помощью нейронных сетей. Предсказание стоимости ноутбуков по их характеристикам.

Отчёт по проекту по дисциплине
Профориентационный семинар "Введение в специальность"
студента образовательной программы бакалавриата
"Прикладная математика"

Студент
Морозов Д.С.

Руководитель ВКР
Профессор
В.Ю. Попов

**Москва 2024г.**

# Содержание

# Аннотация

В рамках данного проекта разрабатывалась модель нейронной сети для решения задачи регрессии с использованием языка программирования Python и датасета «Laptop Price»[9]. Была создана модель, способная прогнозировать цены на ноутбуки в зависимости от набора их характеристик. Средняя абсолютная ошибка на тестовых данных составила 175 евро при средней цене ноутбуков в 1123 евро. Это неплохой показатель, учитывая небольшой размер датасета и большое количество характеристик.

Для достижения полученного результата использовались следующие методы:

- Тщательный анализ и предварительная обработка данных, включающая упрощение их структуры и преобразования категориальных значений в числовые.
- Использование такой модели нейронной сети для регрессии, как многослойный персептрон, из библиотеки Keras языка Python (Sequential())[7].
- Оптимизация архитектуры сети путём добавления слоя нормализации входных данных, а также экспериментального подбора количества нейронов и слоёв. Была выбрана конфигурация с 300 нейронами на входном слое, 1200 и 200 нейронами на первом и втором скрытом слое, соответственно, 1 нейроном на выходном слое.

- Применение L2-регуляризации и Dropout для предотвращения переобучения и улучшения обобщающей способности модели[3]

# Введение

Регрессионный анализ играет большую роль в современном мире, поскольку он позволяет прогнозировать различные числовые значения на основе имеющихся данных. Это важно для множества областей, таких как финансы, медицина, маркетинг, торговля, промышленность и многое другое. Существует масса методов решения задач регрессии, в частности с применением машинного обучения. В рамках данного проекта будет изучаться способ, использующий нейронную сеть для выявления закономерностей в данных и дальнейшего предсказания нужного значения. Для исследования был выбран датасет «Laptop Price» с сайта www.kaggle.com[9], содержащий конфигурации и цены ноутбуков. Весь проект выполнялся на языке программирования Python, в среде разработки Google Colaboratory.

Целью данного проекта является разработка модели нейронной сети, с помощью которой возможно спрогнозировать цены на ноутбуки в зависимости от набора их характеристик с неплохой точностью.

Для достижения поставленной цели был выдвинут ряд задач, требующих последовательного решения:

1. Изучить принципы и современные методы регрессионного анализа с помощью нейросетей и ознакомиться со способами их реализации на языке Python.

2. Выбрать наиболее подходящий метод, основанный на нейросетевых технологиях, и подробнее изучить его.

3. Провести предварительный анализ и обработку выбранных данных для дальнейшей работы с ними.

4. Разработать нейронную сеть согласно выбранной модели для предсказания цен на ноутбуки на основе обработанных данных.

5. Оценить результат работы созданной модели.

# 1 Теоретическая часть

В данной главе подробно описывается проблематика проекта, объясняются основные понятия, а также обозреваются технологии, методы и принципы, позволяющие достичь поставленную в проекте цель – создать модель нейронной сети, способной решить задачу регрессии.

## 1.1 Основы нейросетей в задачах регрессии

Дадим определение регрессии: это статистический метод анализа данных, который используется для изучения отношения между зависимой (целевой) переменной и одной либо несколькими независимыми переменными. Главная цель регрессионного анализа состоит в предсказании значений целевой переменной на основе известных значений независимых переменных[5].

В рамках данного проекта для решения задачи регрессии методами машинного обучения выбраны именно нейронные сети, так как они помогают устанавливать довольно сложные закономерности в данных, состоящих из множества переменных, а также обладают возможностью гибкой настройки под конкретные ситуации для достижения хорошей точности модели.

Перейдём непосредственно к теоретической стороне разработки модели нейросети. Наиболее распространённым типом нейронной сети для регрессии является многослойный персептрон (MLP - multilayer perceptron). Он состоит из нескольких слоёв, включая входной, скрытые и выходной, которые последовательно соединены между собой. На каждом слое находятся нейроны, связанные между собой синаптическими весами. На входном слое нейроны принимают предварительно обработанные значения признаков (независимые переменные) и передают данные на скрытые слои, расположенные между входным и выходным. В скрытых слоях происходит вычисление линейных комбинаций входов и активация нейронов. Количество и раз-

мер скрытых слоёв влияет на сложность модели и её способность аппроксимировать сложные зависимости между данными. Выходной слой содержит всего один нейрон, который и выдаёт предсказанное непрерывное числовое значение целевой переменной[8].

Для нелинейности в сети в каждом нейроне есть функция активации, которая преобразует значения данных согласно определённому принципу. В задачах регрессии обычно используется ReLU (rectifier linear unit) - формула 1, так как именно она позволяет нейросети работать с нелинейными зависимостями[7].

$$f(x) = \max(0, x) \tag{1}$$

В процессе обучения используется алгоритм обратного распространения ошибки. Он заключается в последовательном прохождении вперёд (предсказание) и обратно (обновление весов с учётом ошибки) по нейронной сети. Для корректировки весов минимизируется выбранная функция потерь с помощью оптимизатора. В качестве функции потерь в задачах регрессии используют среднеквадратичную ошибку (MSE – mean squared error) - формула 2, а среди эффективных оптимизаторов выбирают Adam, который помогает сходиться к оптимальному решению с высокой скоростью[8].

$$MSE = \frac{1}{N}\sum_{n=1}^{N}(y_i - \hat{y}_i)^2 \quad (2)$$

## 1.2 Этапы разработки нейронной сети

Для реализации регрессионного анализа с помощью нейронной сети можно выделить следующие этапы разработки модели:

1. Первоначальный анализ и предобработка данных. Аналогично другим методам машинного обучения, нейронная сеть требует тщательной предварительной обработки входных данных для дальнейшей работы. Необходимо перевести все категориальные значения в численный вид, данные не должны содержать пропущенных значений, требуется масштабирование значений либо добавление слоя нормализации в архитектуру нейронной сети.

2. Построение архитектуры нейронной сети. Подбирается количество скрытых слоёв и количество нейронов на каждом слое, а также функция активации. Необходимо создать такую архитектуру, чтобы её сложности хватило для определения непростых взаимосвязей в данных.

3. Определение функции потерь и оптимизатора.

4. Обучение нейронной сети. Сначала данные разделяются на обучающую и тестовую выборку, затем сеть обучается на обучающем наборе до достижения

заданного количества эпох.

5. Оценка результатов работы модели. После завершения обучения проводится проверка нейронной сети на тестовом наборе, смотрятся метрики качества модели, например средняя абсолютная ошибка (MAE – mean absolute error) между предсказанным и истинным значением.

6. Корректировка параметров модели. Если после оценки результатов есть необходимость улучшить качество предсказаний нейросети, можно пробовать менять количество слоёв, количество нейронов или количества эпох обучения и снова смотреть на результат.

## 1.3 Переобучение и борьба с ним

Однако на 5-ом и 6-ом этапах разработки нейросети может возникнуть следующая проблема: с каждой эпохой обучения ошибка модели на валидных данных (те данные, на которых проверяется точность непосредственно в процессе обучения после каждой эпохи) будет только расти вместо того, чтобы уменьшаться вместе с ошибкой на тренировочных данных. Это может быть явным признаком переобучения нашей модели, то есть нейросеть слишком сильно подстраивается к тренировочному набору, заучивая её специфические особенности, которые не обобщаются на все данные, на тестовую выборку. Чтобы устранить явление переобучения можно упро-

стить архитектуру нашей нейронной сети и уменьшить количество эпох обучения, но это неизбежно приведёт к потери точности нашей модели. Поэтому для борьбы с переобучением существуют специальные методы: регуляризация и dropout.

Регуляризация – способ, который добавляет численный штраф в функцию потерь за сложность модели, что помогает ограничить и контролировать её веса. Предпочтительным вариантом является L2 регуляризация (Ridge регуляризация) – добавляет к функции потерь квадрат суммы весовых коэффициентов, делая нейросеть устойчивее к переобучению[7].

Dropout – метод, заключающийся в случайном исключении (отключении) части нейронов в процессе обучения модели с заданной вероятностью. Во время каждой итерации происходит обучение отдельной новой «разрежённой» модели без выбранных нейронов, и в результате такая нейросеть обладает улучшенной обобщающей способностью[7].

## 1.4 Технологический стек

Определим технологический стек, который в дальнейшем будет использоваться при реализации всего проекта. Для этого был выбран язык программирования Python, так как он обладает рядом существенных преимуществ перед другими языками для решения поставленной задачи регрессии. Python довольно прост в изучении и работе, обладает простым синтаксисом, в свободном доступе имеется огромное

количество информационных ресурсов, связанных с ним. Решающим фактором при выборе стало наличие многочисленных библиотек, в частности для машинного обучения, которые значительно расширяют стандартные возможности языка и гораздо упрощают работу с ним. В качестве среды разработки был выбран онлайн сервис Google Colaboratory (Colab), созданный на основе Jupyter Notebook и позволяющий писать и запускать код прямо из браузера, без необходимости устанавливать дополнительные программы на персональный компьютер. Среди достоинств этого сервиса можно выделить то, что в нём предустановлены большинство популярных библиотек для языка Python, он обладает хорошими вычислительными мощностями, а все данные хранятся в облачном хранилище Google Disk.

Теперь изучим функциональные возможности языка Python, которые позволят реализовать все вышеописанные идеи. Для анализа и обработки данных воспользуемся популярнейшими библиотеками Pandas и Numpy. Они позволяют выполнять простые действия с датасетом: посмотреть на его срез, получить характеристику каждого столбца, вывести общую статистику по данным в базе, выполнять математические действия с ними и многое другое[4][2].

Для удобной визуализации данных и полученных результатов будем использовать библиотеку Matplotlib. Кроме этого, для работы с данными, например для кодирования категориальных переменных, обратимся к не менее известной библиотеке Scikit-learn, обладающей большим функционалом для машинного обучения.[6] Нако-

нец, для непосредственного создания нейронной сети, решающей задачу регрессии, будем пользоваться библиотекой Keras, основанной на платформе TensorFlow. Такой выбор обусловлен наличием в Keras класса Sequential(), с помощью которого удобно, просто и наглядно создавать нужную нам последовательную модель (многослойный персептрон). Здесь есть все необходимые нам настраиваемые параметры нейронной сети: количество слоёв, количество нейронов, среднеквадратичная функция потерь, оптимизатор Adam, методы борьбы с переобучением – регуляризация и dropout, функция активации ReLU, количество эпох обучения, метрика средней абсолютной ошибки. Так же данный подход даёт возможность добавить в архитектуру разрабатываемой модели слой нормализации входных данных, что исключает необходимость предварительной нормализации или стандартизации данных и повышает точность предсказываемых значений.[7]

# 2 Практическая часть

В данной главе будет описан сам процесс решения поставленной в проекте задачи регрессии, то есть будут разобраны все этапы непосредственной разработки необходимой модели нейронной сети на языке Python в среде разработки Google Colab.

## 2.1 Подготовка к написанию кода

Сперва было необходимо найти подходящий датасет для регрессионного анализа. Поиск выполнялся на популярном в сфере анализа данных и машинного обучения сайте www.kaggle.com, который является хранилищем большого количества баз данных. Для этого сначала нужно перейти в раздел «Datasets», а затем в поисковой строке «Search dataset» ввести запрос «Regression». В результате будет представлено большое количество датасетов, которые могут использоваться для решения задач регрессии. Среди этого множества был выбран датасет «Laptop Price»[9]. На его странице сразу можно увидеть краткое описание данных, хранящихся в столбцах таблицы: 11 различных характеристик ноутбуков (независимые переменные) и столбец цен на них (целевая переменная), подробный обзор будет далее.

Скачав файл laptop_price.csv с датасетом, загружаем его на Google Disk и подключаем Google Disk к блокноту в Google Colab, в котором будем реализовывать проект. Теперь можно удобно работать с выбранной базой данных.

Настроив среду разработки, можно приступать к написанию кода. С самого начала с помощью import импортируем необходимые библиотеки: pandas, numpy, matplotlib. Из sklearn.preprocessing импортируем метод OneHotEncoder для кодировки категориальных данных (более подробно далее). Из библиотеки tensorflow.keras импортируем описанные в теоретической части методы: Sequential – для создания модели многослойного персептрона, Dropout и regularizers – для борьбы с переобуче-

нием нейросети, Normalization – для создания слоя нормализации входных данных, оптимизатор Adam, layers – для настройки архитектуры нейронной сети.

## 2.2 Предварительный анализ данных

Начнём анализ и обработку данных для последующей корректной работы разрабатываемой модели нейронной сети для регрессии. С помощью функции .read_csv из pandas открываем загруженный ранее датасет laptop_price.csv и называем его data. С помощью методов .head(10) и .info выводим первые 10 строчек (рисунок 1) и характеристику столбцов базы data (таблица 1), соответственно[4].

| | laptop_ID | Company | Product | TypeName | Inches | ScreenResolution | Cpu | Ram | Memory | Gpu | OpSys | Weight | Price_euros |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | Apple | MacBook Pro | Ultrabook | 13.3 | IPS Panel Retina Display 2560x1600 | Intel Core i5 2.3GHz | 8GB | 128GB SSD | Intel Iris Plus Graphics 640 | macOS | 1.37kg | 1339.69 |
| 1 | 2 | Apple | Macbook Air | Ultrabook | 13.3 | 1440x900 | Intel Core i5 1.8GHz | 8GB | 128GB Flash Storage | Intel HD Graphics 6000 | macOS | 1.34kg | 898.94 |
| 2 | 3 | HP | 250 G6 | Notebook | 15.6 | Full HD 1920x1080 | Intel Core i5 7200U 2.5GHz | 8GB | 256GB SSD | Intel HD Graphics 620 | No OS | 1.86kg | 575.00 |
| 3 | 4 | Apple | MacBook Pro | Ultrabook | 15.4 | IPS Panel Retina Display 2880x1800 | Intel Core i7 2.7GHz | 16GB | 512GB SSD | AMD Radeon Pro 455 | macOS | 1.83kg | 2537.45 |
| 4 | 5 | Apple | MacBook Pro | Ultrabook | 13.3 | IPS Panel Retina Display 2560x1600 | Intel Core i5 3.1GHz | 8GB | 256GB SSD | Intel Iris Plus Graphics 650 | macOS | 1.37kg | 1803.60 |
| 5 | 6 | Acer | Aspire 3 | Notebook | 15.6 | 1366x768 | AMD A9-Series 9420 3GHz | 4GB | 500GB HDD | AMD Radeon R5 | Windows 10 | 2.1kg | 400.00 |
| 6 | 7 | Apple | MacBook Pro | Ultrabook | 15.4 | IPS Panel Retina Display 2880x1800 | Intel Core i7 2.2GHz | 16GB | 256GB Flash Storage | Intel Iris Pro Graphics | Mac OS X | 2.04kg | 2139.97 |
| 7 | 8 | Apple | Macbook Air | Ultrabook | 13.3 | 1440x900 | Intel Core i5 1.8GHz | 8GB | 256GB Flash Storage | Intel HD Graphics 6000 | macOS | 1.34kg | 1158.70 |
| 8 | 9 | Asus | ZenBook UX430UN | Ultrabook | 14.0 | Full HD 1920x1080 | Intel Core i7 8550U 1.8GHz | 16GB | 512GB SSD | Nvidia GeForce MX150 | Windows 10 | 1.3kg | 1495.00 |
| 9 | 10 | Acer | Swift 3 | Ultrabook | 14.0 | IPS Panel Full HD 1920x1080 | Intel Core i5 8250U 1.6GHz | 8GB | 256GB SSD | Intel UHD Graphics 620 | Windows 10 | 1.6kg | 770.00 |

Рис. 1: Первые 10 строк исходного датасета

Датасет включает в себя 1303 строк, не содержит пропущенных значений (типа Null) и состоит из следующих столбцов: laptop_ID – уникальный номер ноутбука в таблице (int), Company – название компании-производителя (str), Product – название

| RangeIndex: 1303 entries, 0 to 1302 | | | |
|---|---|---|---|
| # | Column | Non-Null Count | Dtype |
| 0 | laptop_ID | 1303 non-null | int64 |
| 1 | Company | 1303 non-null | object |
| 2 | Product | 1303 non-null | object |
| 3 | TypeName | 1303 non-null | object |
| 4 | Inches | 1303 non-null | float64 |
| 5 | ScreenResolution | 1303 non-null | object |
| 6 | Cpu | 1303 non-null | object |
| 7 | Ram | 1303 non-null | object |
| 8 | Memory | 1303 non-null | object |
| 9 | Gpu | 1303 non-null | object |
| 10 | OpSys | 1303 non-null | object |
| 11 | Weight | 1303 non-null | object |
| 12 | Price_euros | 1303 non-null | float64 |

Таблица 1: Характеристика столбцов исходного датасета.

модели (str), TypeName – тип ноутбука (str), Inches – диагональ монитора в дюймах (float), ScreenResolution – характеристика дисплея с разрешением (str), Cpu – процессор и его частота (str), Ram – оперативная память (str), Memory – тип памяти и размер (str), Gpu – видеокарта (str), OpSys – операционная система (str), Weight - вес (str), Price_euros – цена ноутбука в евро (int). В таком виде нейросеть не сможет работать с данными, необходимо представить их в численном виде.

Посмотрим на статистику целевой переменной – цены на ноутбуки из столбца Price_euros. Воспользуемся методом .describe()[4], с помощью которого узнаем: минимальная цена – 174, максимальная – 6099, средняя цена по датасету – 1123, стандартное отклонение – 699. Также построим гистограмму распределения цен с помо-

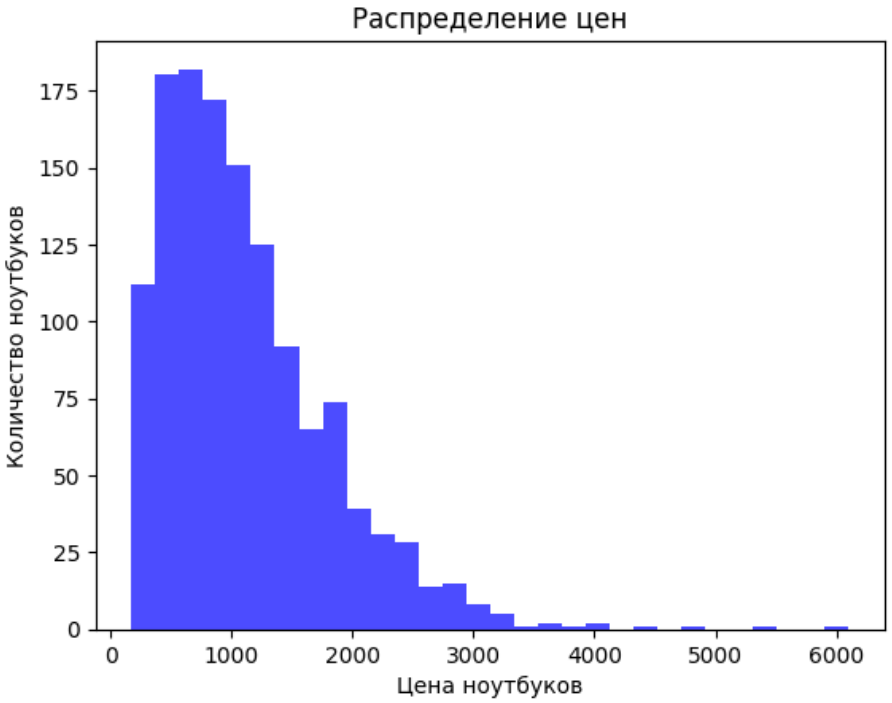


Рис. 2: Гистограмма распределения цен на ноутбуки

щью .hist из matplotlib.pyplot (рисунок 2).

## 2.3 Обработка данных

Первым шагом было решено в столбцах Ram и Weight убрать размерности GB и kg и перевести в тип int и float, соответственно, то есть в численный вид. Перед обработкой остальных категориальных данных нужно понять, какую они имеют структуру (по первым 10 строкам понять это невозможно). Для этого к столбцам Company, Product, TypeName, ScreenResolution, Memory, Cpu, Gpu, OpSys применяем метод .value_counts()[4], который выводит все уникальные значения из столбца и их количество. Пример его работы на рисунке 3.

```
OpSys
Windows 10      1072
No OS             66
Linux             62
Windows 7         45
Chrome OS         27
macOS             13
Mac OS X           8
Windows 10 S       8
Android            2
Name: count, dtype: int64
```

Рис. 3: Пример работы .value_counts()

Проанализировав все результаты выполнения данных команд, было решено оставить столбцы Company, TypeName, OpSys неизменными и в дальнейшим просто перевести их в численный вид, так как они содержат небольшое количество уникальных значений и структура данных в них простая – одно слово. Столбец Product содержит 618 уникальных значений, что очень много относительно общего количества строк в датасете (1303), поэтому его придётся удалить из рассмотрения с помощью метода .drop(). По этой же причине удалим столбец Gpu.

Оставшиеся категориальные столбцы ScreenResolution, Memory и Cpu также содержат немало уникальных значений и обладают сложной составной структурой, но данные в них можно разбить на несколько столбцов, тем самым упростив их. Рассмотрим их обработку по отдельности.

Столбец ScreenResolution разбиваем на 4 столбца: Ips – наличие матрицы Ips (0 – нет, 1 – есть), Touchscreen – наличие сенсора на экране (0 – нет, 1 – есть), res_X – горизонтальная составляющая разрешения, res_Y – вертикальная составляющая

разрешения.

Столбец Cpu разбиваем на 2 столбца: Cpu_name – название модели процессора, Cpu_freq – частота процессора в GHz.

В столбце Memory сначала переводим размерности памяти из TB в GB приписыванием 3-х нулей, чтобы везде была одна величина измерения. Затем, если память складывается из двух составляющих одинакового типа (например 256GB SSD + 256GB SSD), то заменяем эти значения на сумму (в примере – на 512GB SSD). Теперь создадим 4 столбца: SSD, HDD, Flash Storage, Hybrid. И в эти столбцы запишем размер соответствующей памяти. Если какой-либо из типов отсутствует, то устанавливаем значение 0.

Теперь удаляем изначальные необработанные столбцы: ScreenResolution, Cpu, Memory. На рисунке 4 результат проделанной обработки.

| | Company | TypeName | Inches | Ram | OpSys | Weight | Price_euros | Ips | Touchscreen | res_X | res_Y | Cpu_name | Cpu_freq | SSD | HDD | Flash Storage | Hybrid |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Apple | Ultrabook | 13.3 | 8 | macOS | 1.37 | 1339.69 | 1 | 0 | 1600 | 1600 | Intel Core i5 | 2.3 | 128 | 0 | 0 | 0 |
| 1 | Apple | Ultrabook | 13.3 | 8 | macOS | 1.34 | 898.94 | 0 | 0 | 900 | 900 | Intel Core i5 | 1.8 | 0 | 0 | 128 | 0 |
| 2 | HP | Notebook | 15.6 | 8 | No OS | 1.86 | 575.00 | 0 | 0 | 1080 | 1080 | Intel Core i5 | 2.5 | 256 | 0 | 0 | 0 |
| 3 | Apple | Ultrabook | 15.4 | 1 | macOS | 1.83 | 2537.45 | 1 | 0 | 1800 | 1800 | Intel Core i7 | 2.7 | 512 | 0 | 0 | 0 |
| 4 | Apple | Ultrabook | 13.3 | 8 | macOS | 1.37 | 1803.60 | 1 | 0 | 1600 | 1600 | Intel Core i5 | 3.1 | 256 | 0 | 0 | 0 |

Рис. 4: Обработанный датасет

Данные приняли достаточно простой вид. Остаётся закодировать категориальные значения, чтобы они стали численными. Для этого воспользуемся методом OneHotEncoder[6] – он создаёт для каждой уникальной категории данных отдельный столбец и расставляет 0 или 1 в соответствии со значениями в изначальных

столбцах. Таким образом категориальные данные примут бинарный вид.[1]

На этом обработка данных заканчивается и можно приступать к разработке самой нейронной сети.

## 2.4 Создание и обучение модели нейронной сети

Перед тем как начать работу над моделью, нужно разбить имеющиеся данные на тренировочную и тестовую выборки. Для создания тренировочного набора был использован метод .sample() библиотеки Pandas с аргументом frac=0.85 – 85% от всего датасета. Тестовый набор получаем удалением тренировочного из исходного датасета методом .drop(). В свою очередь, обе выборки были разбиты на две составляющие: X – набор независимых признаков, y – значения целевой переменной. Сразу важно заметить, что выбранный способ разделения данных каждый раз случайно выбирает строки, из-за чего при неоднократном обучении одной и той же модели результаты ошибки могут незначительно отличаться друг от друга.[8]

Перейдём к построению архитектуры нейронной сети для регрессии на основе знаний, полученных в первой главе «Теоретическая часть». Сразу было решено сделать два скрытых слоя, так как выбранный датасет содержит много разнообразных признаков. С помощью Sequential() создаётся модель многослойного персептрона. Сначала идёт слой нормализации, созданный функцией Normalization(), так как на вход модели мы подаём ненормализованные и нестандартизированные данные. Да-

лее идут входной, первый скрытый, второй скрытый слои с функцией активации ReLU и выходной слой из одного нейрона с линейной функцией активации. Изначально количество нейронов было 64-256-64 (соответственно описанным слоям), но в дальнейшем было изменено. Для компиляции модели выбрана функция ошибок MSE, оптимизатор Adam, метрика MAE.[7]

После построения архитектуры приступаем к обучению нейросети. Для этого используем .fit(), где validation_split=0.1 – данные для валидации составляют 10% от тренировочной выборки, epochs=200 – количество эпох обучения. На рисунке 5 изображён процесс обучения.

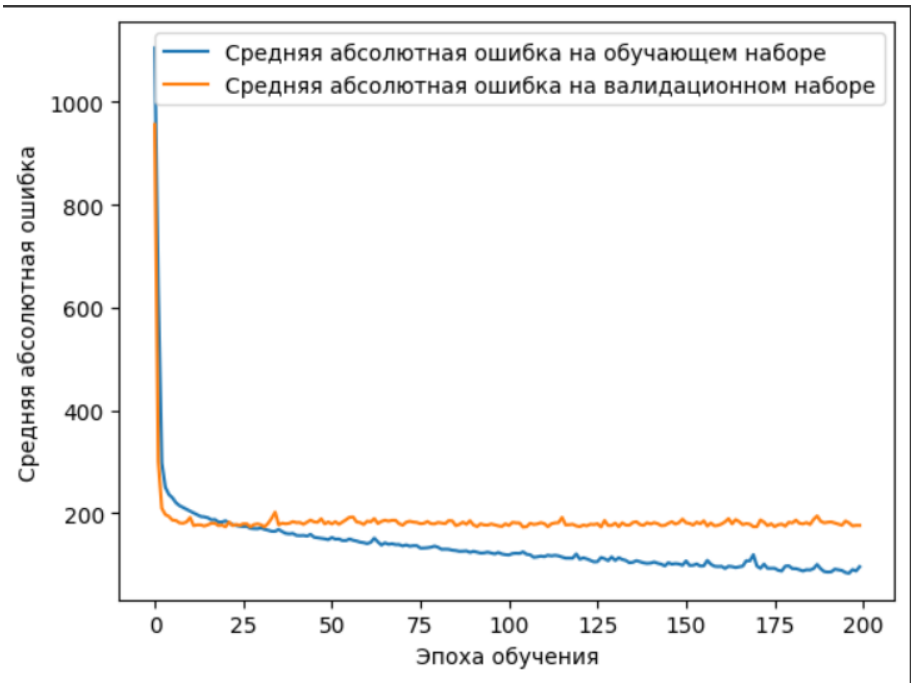


Рис. 5: Обучение первой модели

Видно, что ошибка на валидационном наборе не уменьшается и равна пример-

но 200. С помощью метода .evaluate()[7] получаем среднюю абсолютную ошибку на тестовых данных, равную 210.

Попробуем улучшить качество модели. Смысла делать больше эпох обучения нет, поэтому просто увеличим число нейронов. Была выбрана такая конфигурация: 300-1200-200. Обучив данную модель таким же способом, как и первую, получился похожий график, но при проверке на тестовых данных ошибка составила 2090636416. Это говорит о явном переобучении, от которого постараемся избавиться. Для этого на входной и скрытые слои добавим L2 регуляризацию с аргументом 0.001, а также применим Dropout к входному и первому скрытому слоям с вероятностью 0.5.[3] Итог оптимизации модели представлен на рисунке 6.

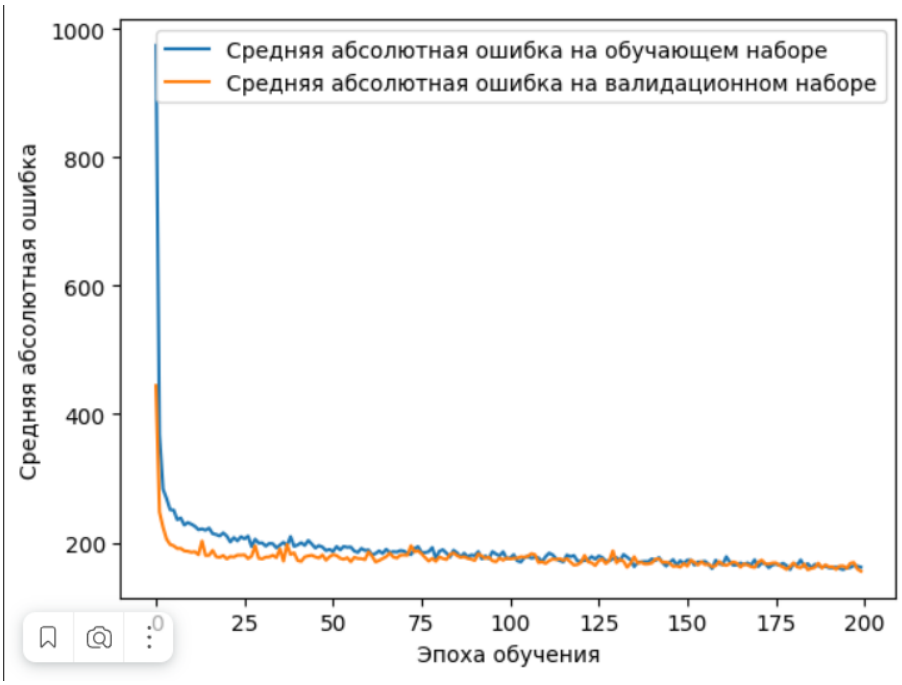


Рис. 6: Обучение оптимизированной модели

Проверив эту модель на тестовом наборе, получаем среднюю абсолютную ошибку 175. Этот результат заметно лучше первой архитектуры.

## 2.5 Оценка результата

Чтобы получить предсказанные значения целевой переменной на основе признаков из тестовой выборки с помощью разработанной модели, применим .predict()[7]. Как сказано ранее, средняя абсолютная ошибка между предсказанными и истинными значениями цен ноутбуков составляет 175 евро, что довольно неплохо при средней цене в 1123 евро и небольшом датасете для обучения с большим количеством признаков (характеристик ноутбуков). Самая маленькая ошибка равна 2 евро, самая большая – 1590 евро.

На рисунке 7 можно видеть, что, в основном, чем меньше ошибка, тем чаще она встречается, небольшие ошибки преобладают над существенными промахами, что является хорошим показателем точности созданной модели. Так же для оценки результата можно построить точечную диаграмму с предсказанными и истинными ценами (рисунок 8).

Здесь так же видно, что большинство пар предсказанных и истинных значений (расположены на одной вертикали) довольно близки друг к другу.

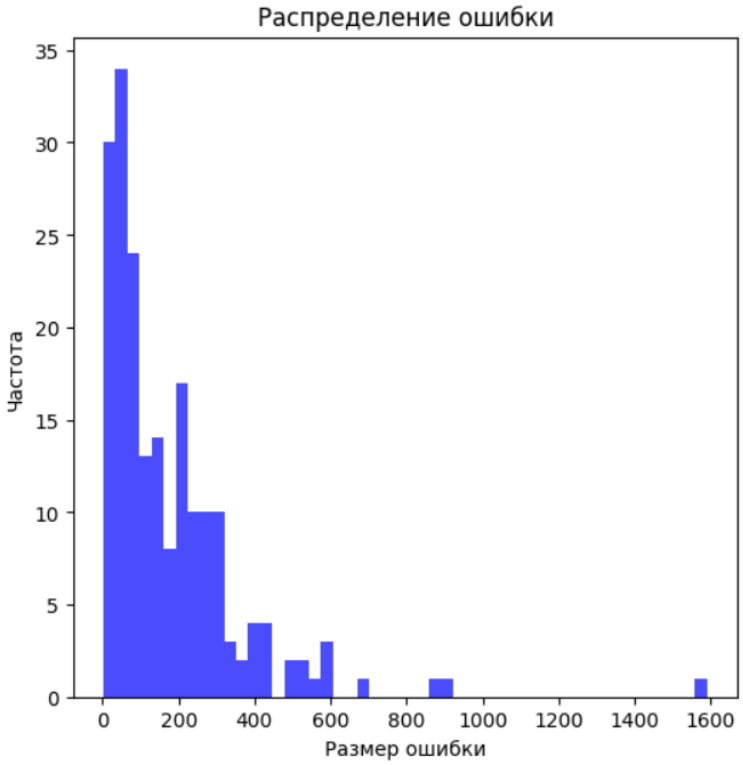


Рис. 7: Гистограмма распределения ошибки

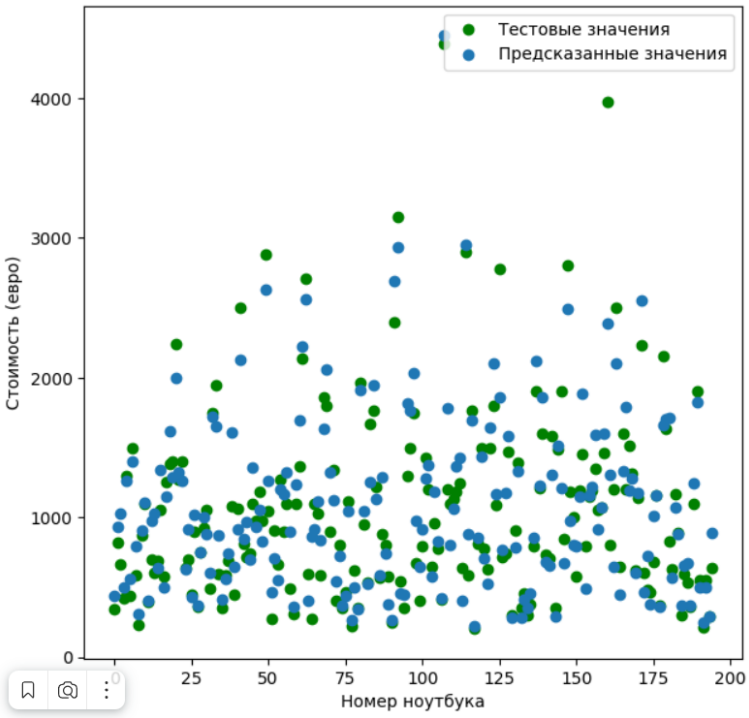


Рис. 8: Предсказанные и истинные значения

# Заключение

В конечном итоге поставленная в проекте цель была достигнута – разработана модель, способная прогнозировать цены на ноутбуки в зависимости от набора их характеристик на основе датасета «Laptop Price» со средней абсолютной ошибкой в 175 евро. В процессе реализации была изучена специфика создания нейронных сетей для регрессии: последовательная модель, настройка архитектуры, функция активации ReLU, функция потерь MSE, оптимизатор Adam, методы предотвращения переобучения - регуляризация и Dropout.

В перспективе созданная модель имеет потенциал быть усовершенствованной в плане точности предсказываемых значений, например, путём усложнения её архитектуры, обучения на более масштабной базе данных и больших вычислительных мощностях, использования других методик обработки данных.

Работа над проектом помогла лучше познакомиться с технологиями машинного обучения, изучить основы разработки и оптимизации моделей нейронных сетей, а также улучшить навыки исследования и обработки данных с помощью возможностей языка Python.

# Список литературы


[1] Aleksandr. *Kaggle Competition Notebook. Base_Line_3_1. Решение задачи регрессии с помощью нейронных сетей.* https://www.kaggle.com/code/lkatran/baseline-3-1. 2021.

[2] *Numpy Documentation.* https://numpy.org/doc/stable/.

[3] *OpenAI. ChatGPT 3.5.* https://chatgpt.com/?oai-dm=1.

[4] *Pandas Documentation.* https://pandas.pydata.org/docs/.

[5] *Regression in machine learning.* https://www.geeksforgeeks.org/regression-in-machine-learning/. 26 февраля 2024.

[6] *Scikit-learn Documentation.* https://scikit-learn.org/stable/index.html.

[7] *TensorFlow Core. Гид. Keras.* https://www.tensorflow.org/guide/keras?hl=ru.

[8] *TensorFlow Core. Руководства. Основы машинного обучения с Keras.* https://www.tensorflow.org/tutorials/keras/regression?hl=ru.

[9] Muhammet Varli. *Kaggle Datasets "Laptop Price".* https://www.kaggle.com/datasets/muhammetvarl/laptop-price. 2020.