ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ БЮДЖЕТНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ»**

**(ФИНАНСОВЫЙ УНИВЕРСИТЕТ)**

Департамент анализа данных

и машинного обучения

***Дисциплина: «Технологии анализа данных и машинного обучения»***

*Направление подготовки: «Прикладная математика и информатика»*

*Профиль: «Анализ данных и принятие решений в экономике и финансах»*

*Факультет информационных технологий и анализа больших данных*

*Форма обучения очная*

*Учебный 2022/2023 год, 7 семестр*

**Курсовая работа на тему:**

«Методы глубокого обучения в задачах прогнозирования объема потребления »

*Выполнил:*

студент группы ПИ20-3

Ленгауэр А. Д.

*Научный руководитель:*

доцент д.ф.-м.н. Золотова Т. В.

**Москва 2022**

**Оглавление**

Введение…………………………………………………………………………...3

Глава 1. Теоретические основы методов глубокого обучения...........................4

* 1. Разбор различных типов нейронных сетей………………………………….4
  2. Обзор основных типов функций активации...................................................6
  3. Различные виды оптимизационных алгоритмов и функций потерь............6
  4. Выбор необходимого для моей работы вида нейронных сетей и данных...8

Глава 2. Практическая часть...................................................................................9

* 1. Предобработка данных.....................................................................................9
     1. Обзор исходных данных...............................................................................9
     2. Преобразование данных для использования в моделях глубокого обучения.......................................................................................................12
     3. Обработка пропущенных значений и выбросов.......................................15
     4. Разделение на тренировочную и тестовую выборки................................16
  2. Обучение модели и анализ данных................................................................17
     1. Построение класса модели нейронной сети..............................................17
     2. Обучение модели на тренировочных данных...........................................17
     3. Анализ результатов обучения.....................................................................18
     4. Построение CNN модели глубокого обучения.........................................21
     5. Обучение новой модели на тренировочных данных................................23
     6. Анализ результатов обучения новой модели............................................24
     7. Построение 3 модели глубоко обучения...................................................25
     8. Обучение 3 модели на тренировочных данных........................................25
     9. Анализ результатов обучения 3 модели....................................................26

1. Выводы..............................................................................................................28
   1. Анализ результатов работы............................................................................28
   2. Общие выводы по использованию методов глубокого обучения для прогнозирования объема потребления...........................................................29

Список использованных источников...................................................................29

**Введение**

В наши дни ходит много разговоров про нейронные сети и методы глубокого обучения. Эти понятия пробрались почти во все сферы нашей жизни. Сейчас нейронные сети можно встретить в смартфонах, телевизорах и даже зубных щетках.

Итак, я выбрал тему прогнозирования объема потребления при помощи глубоких нейронных сетей. Эта тема актуальна по нескольких причинам.

* Во-первых, объем потребления является важным экономическим показателем, который определяет спрос на товары и услуги в различных отраслях экономики. Это в свою очередь влияет на производство, цены, занятость и многие другие аспекты жизни общества. Поэтому прогнозирование объема потребления является ключевой задачей для бизнеса, государства и исследовательской общественности.
* Во-вторых, с появлением больших данных и вычислительных ресурсов стали доступны методы глубокого обучения, которые показали свою эффективность во многих областях, включая прогнозирование. Эти методы позволяют автоматически извлекать признаки из данных, учитывать нелинейные зависимости между ними и создавать сложные модели, которые могут достичь высокой точности прогнозирования.
* В-третьих, существует множество задач прогнозирования объема потребления, которые можно решать с помощью методов глубокого обучения. Например, это могут быть задачи прогнозирования продаж в розничной торговле, объема заказов на транспортных компаниях, числа кликов на рекламные объявления в интернете и т.д.
* В-четвертых, применение методов глубокого обучения в задачах прогнозирования объема потребления может привести к значительному экономическому эффекту для бизнеса и государства. Например, точное прогнозирование спроса может позволить оптимизировать запасы и производственные мощности, снизить затраты на складирование и транспортировку товаров, улучшить качество обслуживания клиентов и так далее.

Таким образом, проблема прогнозирования объема потребления остается актуальной и важной задачей, а применение методов глубокого обучения может привести к значительному улучшению результатов прогнозирования и экономическому эффекту.

Цель работы по теме заключается в исследовании и разработке методов прогнозирования объема потребления с использованием глубокого обучения, с целью повышения точности прогнозов и экономического эффекта для бизнеса. Точные прогнозы объема потребления необходимы для принятия эффективных решений в бизнесе, таких как планирование производства, закупки и реализации товаров. Разумеется бизнес – это не единственное направление, где полезно предсказание спроса и объема потребления. Это также может быть полезно государству, например прогнозирование потребления электроэнергии или газа. Помимо этого, можно предугадывать спрос на новую музыку, фильмы, видео и так далее. Также прогнозирование потребления может включать в себя расчет загруженности дорог, так как это тоже в своем роде потребление услуг общественной инфраструктуры. Обобщая, можно сказать, что цель работы заключается не только в прогнозировании потребления, но и в исследовании возможностей глубокого обучения для решения конкретных задач. Итак, целью работы является разработка нейронной сети для прогнозирования объемов потребления.

В моей работе я буду прогнозировать объем потребления продукции Adidas в Америке. Такие предсказания могут быть очень полезны для бизнеса, так как знание будущих продаж позволят более точно настроить объем производства товаров, маркетинг компании и много других параметров.

Для достижения данной цели необходимо решить следующие задачи:

* Изучение существующих методов прогнозирования объема потребления.
* Изучение принципов работы и реализации глубоких нейронных сетей, используемых для прогнозирования.
* Поиск данных, необходимых для прогнозирования объема потребления, и их предварительная обработка.
* Разработка моделей глубокого обучения для прогнозирования объема потребления.
* Исследование разработанных моделей по метрикам точности и экономического эффекта.
* Оценка применимости разработанных моделей в реальных условиях.

**Глава 1**

* 1. **Разбор различных типов нейронных сетей.**

Существует множество различных видов нейронных сетей. Каждая из них лучше приспособлена к конкретным специфическим задачам и данным. Поэтому важно использовать правильный тип нейронных сетей при решении своей задачи. Я могу привести пример нескольких типов нейронных сетей.

**Перцептрон**

Перцептрон - это простейшая форма искусственной нейронной сети. Он состоит из одного или нескольких слоев нейронов, которые могут обрабатывать только бинарные данные (1 или 0). Каждый нейрон перцептрона принимает на вход набор значений, умножает их на соответствующие им веса и складывает результаты. Затем полученная сумма подается на функцию активации, которая определяет, будет ли нейрон активирован или нет. Если нейрон активирован, то его выходное значение равно 1, иначе - 0.

Перцептроны могут использоваться для решения задач бинарной классификации, когда необходимо определить, принадлежит ли объект одному из двух классов объектов. Для этого необходимо обучить перцептрон, подавая ему на вход обучающие примеры с известной меткой класса и корректируя веса связей между нейронами в процессе обучения.

Перцептроны были первоначальным шагом в развитии искусственных нейронных сетей. Он является одной из наиболее простых и понятных архитектур. Но в настоящее время перцептроны редко используются в практических задачах, так как их способности крайне ограничены в сравнении с более сложными архитектурами.

**Рекуррентная нейронная сеть**

RNN способна учитывать последовательность данных и предсказывать будущие значения на основе предыдущих. RNN отличается от других нейронных сетей тем, что имеет внутреннюю память, которая позволяет сохранять информацию о предыдущих входах и использовать эту информацию для обработки последующих входов. RNN может быть использована для различных задач, связанных с последовательными данными, таких как прогнозирование временных рядов, распознавание речи, обработка естественного языка и т.д. При упоминании RNN, также упоминают такое понятие как LSTM. Оно означает, что нейронная сеть может запоминать нужную информацию и забывать ненужную в процессе обучения.

**Сверточная нейронная сеть**

Сверточная нейронная сеть – это тип нейронной сети, который лучше всего подходит для обработки изображений или видео, то есть данных с большим количеством признаков и большой размерностью. Обучение CNN основано на обратном распространении ошибки, которое помогает выбрать правильные веса нейронной сети. CNN состоит из нескольких слоев, включая сверточные слои, пулинг слои и полносвязные слои. Общий смысл такого типа нейронной сети в том, что он позволяет сократить объем обрабатываемой информации с минимальными потерями, ускорив тем самым процесс обучения.

**Многослойный перцептрон**

В MLP каждый слой состоит из нейронов, которые используются для преобразования входных данных. Входные данные проходят через несколько слоев, в каждом из которых происходит линейная комбинация входных данных и весов нейронов, а затем полученный результат подвергается нелинейной функции активации, например, сигмоиде или ReLU. Такой класс сетей неплохо решает задачи классификации или регрессии. MLP может быть использована вместе с другими архитектурами нейронных сетей, такими как CNN и RNN, для создания более сложных моделей и решения более сложных задач.

* 1. **Обзор основных типов функций активации**

Для начала нужно понять, что такое функция активации. Функция активации — это нелинейная функция, которая применяется к выходу нейрона, чтобы добавить нелинейность в нейронную сеть. Вот обзор некоторых основных типов функций активации, которые широко используются в нейронных сетях:

* Сигмоид - это функция активации, которая выводит значения между 0 и 1, что делает ее полезной для решения задач классификации. Однако, сигмоид имеет проблему градиентного затухания и обычно используется только в последнем слое для бинарной классификации.
* ReLU - это функция активации, которая выводит 0 для всех отрицательных значений, а для положительных значений - она выводит само значение. Она очень эффективна и широко используется во многих моделях глубокого обучения.
* Leaky ReLU - это модификация ReLU, которая добавляет маленький отрицательный наклон для отрицательных значений, чтобы избежать проблемы "мёртвых" нейронов.
* Гиперболический тангенс - это функция активации, которая выводит значения между -1 и 1. Она полезна для задач классификации и регрессии, но также имеет проблему градиентного затухания.
* Softmax - это функция активации, которая используется в последнем слое для задач многоклассовой классификации. Она выводит вероятности каждого класса и обеспечивает их сумму равной 1.
* Swish - это новая функция активации, которая является комбинацией сигмоиды и ReLU и была представлена недавно. Swish быстро становится популярной и показывает хорошие результаты во многих задачах.

Выбор функции активации зависит от конкретной задачи, архитектуры нейронной сети и даже личных предпочтений и опыта исследователя.

* 1. **Различные виды оптимизационных алгоритмов и функций потерь**

Оптимизационные алгоритмы и функции потерь являются двумя ключевыми компонентами нейронных сетей, которые используются для обучения моделей машинного обучения. Вот несколько различных видов оптимизационных алгоритмов и функций потерь:

Оптимизационные алгоритмы:

* Градиентный спуск (Gradient Descent) - это наиболее распространенный алгоритм оптимизации, который используется для обучения нейронных сетей. Он использует градиент функции потерь для обновления весов сети и минимизации функции потерь.
* Стохастический градиентный спуск (Stochastic Gradient Descent) - это улучшенная версия градиентного спуска, которая выбирает случайные подмножества данных (так называемые "mini-batch'и") для вычисления градиента и обновления весов.
* Adam (Adaptive Moment Estimation) - это алгоритм оптимизации, который комбинирует идеи из градиентного спуска и стохастического градиентного спуска. Он использует два оценщика момента, которые адаптивно изменяют скорость обучения в зависимости от изменения градиента.
* Adagrad (Adaptive Gradient) - это алгоритм оптимизации, который адаптивно изменяет скорость обучения для каждого веса, исходя из истории градиента.
* И так далее

Функции потерь:

* Средняя квадратичная ошибка (Mean Squared Error) — это функция потерь, которая измеряет среднеквадратичную разницу между прогнозируемыми и фактическими значениями. Она широко используется для задач регрессии.
* Кросс-энтропия (Cross-entropy) — это функция потерь, которая используется для задач классификации. Она измеряет расхождение между прогнозируемыми и фактическими значениями в виде вероятностей классов.
* Хи-квадрат (Chi-square) — это функция потерь, которая используется для сравнения наблюдаемых и ожидаемых частот в задачах статистики.
* Huber Loss — это функция потерь, которая широко используется в задачах регрессии и является более устойчивой к выбросам, чем средняя квадратичная ошибка.
* Средняя абсолютная ошибка (MAE) – она вычисляется путем нахождения суммы абсолютных разниц между прогнозируемыми и фактическими значениями, а затем деления этой суммы на количество наблюдений.
* Коэффициент детерминации (R-squared (R2)) - является статистической метрикой, которая используется для оценки качества модели регрессии. Он показывает, насколько хорошо модель соответствует данным. Коэффициент детерминации R2 измеряет долю дисперсии зависимой переменной, которая объясняется линейной регрессией. Другими словами, R2 показывает, насколько точно модель может предсказывать значения зависимой переменной по заданным независимым переменным. R2 находится в диапазоне от 0 до 1. Значение 0 означает, что модель не объясняет изменчивости в данных, а 1, что модель полностью объясняет изменчивость в данных.
* И так далее
  1. **Выбор необходимого для моей работы вида нейронных сетей и данных.**

Исходя из всего выше сказанного, я выбираю тип MLP для своей работы. Выбор пал на этот тип по нескольким причинам. Во-первых, он достаточно универсален и может применяться ко множеству различных задач. Во-вторых, данный тип прост в понимании и исполнении. В-третьих, MLP модель имеет широкие возможности по оптимизации и точечной настройке параметров. Помимо этого, структуру MLP сети также легко менять для оптимизации работы модели. К плюсам такого вида нейронных сетей можно отнести то, что он может выявлять сложные нелинейные зависимости между входными данными и выходным значением. Если зависимость между объемом потребления и факторами, влияющими на него, не может быть описана простыми линейными моделями, MLP модель может использоваться для изучения этих зависимостей и предсказания объема потребления на основе этих данных. Кроме этого, MLP прекрасно работает с большим количеством данных и может быстро обучаться на этих признаках. MLP модель позволяет легко обрабатывать как числовые, так и категориальные данные. Это позволяет ей использовать различные типы данных для прогнозирования объема потребления. То есть обобщая, можно сказать, что MLP модель может быть легко настраиваема и может иметь гибкую архитектуру, что позволяет ей адаптироваться к различным задачам прогнозирования, а в совокупности с возможностью быстрой обработки больших данных, эта модель становится одним из лучших кандидатов на выбор для выполнения целей моей курсовой работы.

Но было бы не корректно использовать только один вид нейронных сетей. Так как, несмотря на все плюсы MLP модели, она может оказаться не самым эффективным и дееспособным выбором. Поэтому в качестве альтернативы я выбрал CNN модель. Обычно, сверточные нейронные сети (CNN) используются для решения задач компьютерного зрения, таких как классификация изображений и распознавание объектов на изображениях. Однако, в некоторых случаях, CNN модель также может быть использована для прогнозирования объема потребления, так как она способна анализировать и предсказывать временные ряды. Соответственно CNN модель может быть использована для изучения шаблонов и зависимостей между этими значениями, что может помочь в прогнозировании будущих значений. CNN модель может выделять важные признаки из данных. В случае прогнозирования объема потребления CNN модель может выделять важные факторы, влияющие на объем потребления, такие как день, город, категория продукта и так далее. Сверточная сеть тоже может быть точечно настроена под конкретную задачу. Все эти факторы делают CNN модель неплохим выбором для предсказания объема потребления.

Итак, когда я определился с необходимыми мне моделями, я начал выбирать подходящие данные. В качестве исходных данных я выбрал датасет «Adidas Sales DataBase». В этих данных указана информация о продажах товаров Адидас в Америке за 2020–2021 год. Данный датасет был размещен в открытом доступе на одной из крупнейших платформ, под названием Kaggle. Это платформа принадлежит и поддерживается небезызвестной компанией Google.

**Глава 2**

* 1. **Предобработка данных.**
     1. **Обзор исходных данных.**

После импорта необходимых библиотек (рис. 1) и загрузки датасета (рис. 2), мною были отображены исходные данные датасета (рис. 3). При загрузке данных использовалось несколько параметров метода read\_excel библиотеки pandas. Во-первых, это параметр skiprows. Этим параметром я пропускаю первые 4 строки файла excel, так как начало таблицы данных располагается не в первой строчке. Во-вторых, это параметр header = 0. Он используется для того, чтобы сделать первую строчку таблицы заголовками pandas DataFrame. После этого, используя usecols, убирается первый столбец из данных, так как таблица начинается не с первого столбца, соответственно при загрузке первый столбец просто пустой. Именно он и убирается.

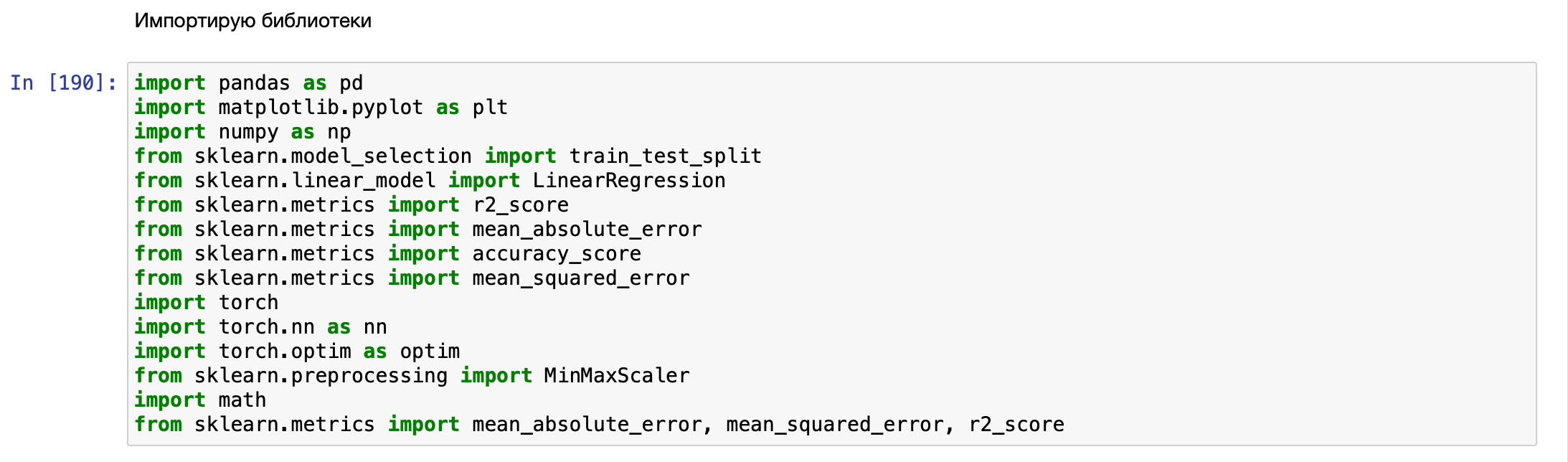


Рис. 1 – Импорт библиотек

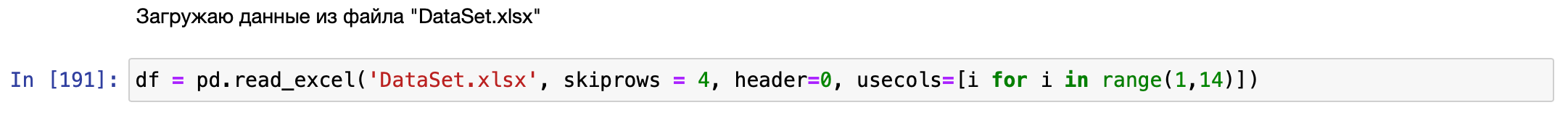


Рис. 2 – Загрузка исходных данных

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рис. 3 – Отображение исходных данных

На рисунке 3 представлены все 13 признаков моего датасета.

* Retailer – это наименование сети магазинов, которые продают продукцию.
* Retailer ID – это уникальный идентификатор сети магазинов.
* Invoice Date – это дата продажи.
* Region – это регион Америки, в котором произошла продажа.
* State - это штат Америки, в котором произошла продажа.
* City - это город, в котором произошла продажа.
* Product – это категория продуктов Адидас, которая была продана.
* Price per unit – Цена за один отдельный товар.
* Units Sold – Количество проданных изделий. Другими словами, это объем потребления. Именно этот признак будет предсказываться.
* Total Sales – Сколько компания получила с продажи изделий.
* Operating Profit – выручка компании.
* Operating Margin – маржа операционной прибыли.
* Sales Method - это метод, которым был куплен товар.

Далее мною были рассмотрены количественные параметры датасета (Рис. 4). Как уже было упомянуто, в моих данных 13 столбцов и 9648 строк. С помощью метода pandas describe(), выводятся основные характеристики числовых признаков, такие как максимальное значение, среднее значение и т.д.

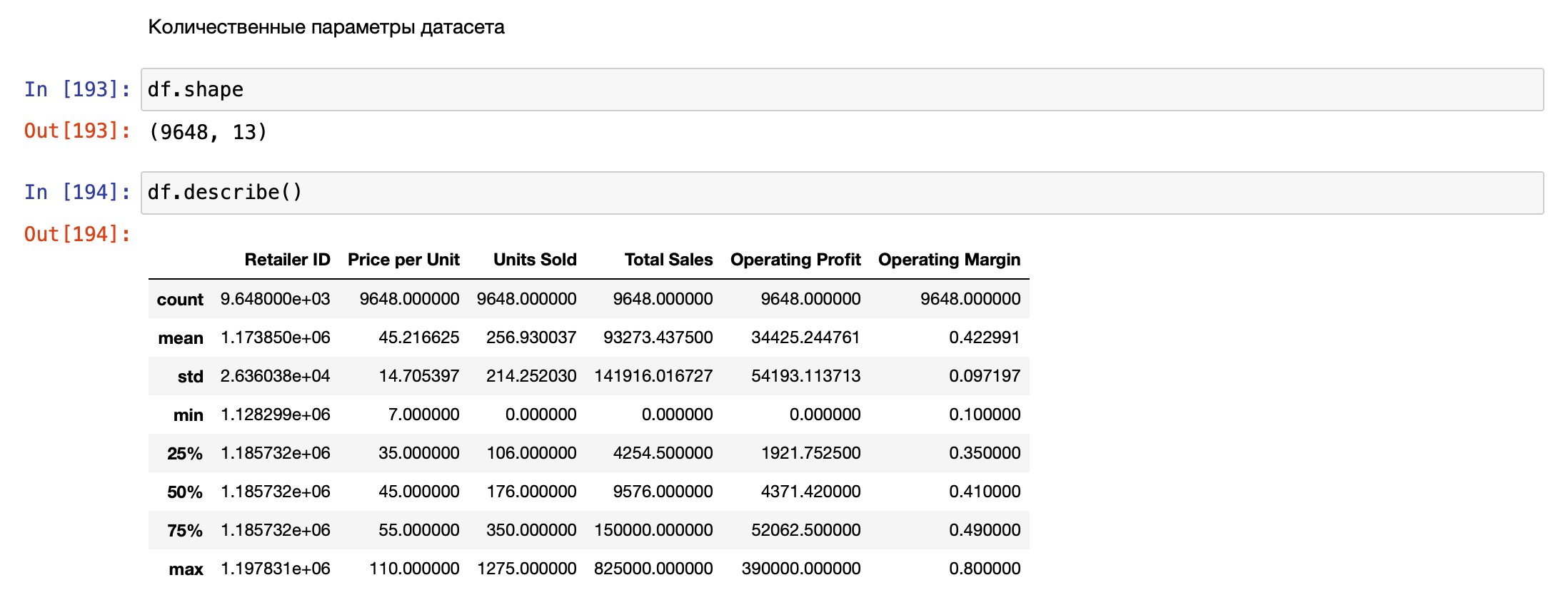


Рис. 4 – Количественные параметры датасета.

Дальше было выяснено количество уникальных значений для каждого признака (Рис. 5) и тип признаков (Рис.6)

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рис. 5 - Количество уникальных значений. Рис. 6 – Тип признаков.

Было найдено количество пропусков датасета (Рис. 7). Выяснилось, что в выбранных данных нет пропусков.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рис. 7 – Количество пропусков.

* + 1. **Преобразование данных для использования в моделях глубокого обучения**

Перед преобразованием данных нужно выяснить как именно будет оно проходить. Так как модель может оперировать только числами, значит нужно представить все данные в виде чисел.

Признак «Retailer» содержит 6 уникальных значений. Значит их можно заменить на числа от 0 до 5. Для этого можно написать собственную функцию, но библиотека pandas предоставляет свой метод для решения этой задачи поэтому было выбрано использовать именно ее. Эта функция называется factorize(). Для выбранного признака она заменяет уникальные значения на цифры (Рис. 8).

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание

Рис. 8 – Преобразование «Retailer».

Следующий идет признак «Region». Он содержит 5 уникальных значений. На нем было решено применить метод get\_dummies () библиотеки pandas (Рис. 9). Этот метод используется для создания фиктивных переменных (dummy variables) на основе категориальных признаков в таблице данных, которые могут быть использованы в анализе данных и машинном обучении.

Фиктивные переменные — это бинарные переменные, которые принимают значение 1, если категориальный признак имеет определенное значение, и 0 в противном случае.

Изображение выглядит как текст, число, Шрифт, линия

Автоматически созданное описание

Рис. 9 – Изменение признака «Region».

Для признака «State» было выбрано использовать метод factorize ().

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рис. 10 – Изменение признака «State».

Для признаков «City» и «Sales Method» также был выбран метод factorize (). (Рис. 11, Рис. 12)

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рис. 11 - Изменение признака «City».

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание

Рис. 12 - Изменение признака «Sales Method».

Для признака «product» был применен метод get\_dummies (). (Рис. 13)

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рис. 13 – Изменение признака «Product».

Признак «Invoice Date» был разделен на три признака: год, месяц, день (Рис. 14).

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рис. 14 – Изменение признака «Invoice Date».

Признак «Retailer ID» было решено удалить, так как он не оказывается сильного влияния на другие данные и, по сути, дублирует «Retailer». (Рис. 15)

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание

Рис. 15 – Удаление признака «Retailer ID».

* + 1. **Обработка пропущенных значений и выбросов.**

Ранее было выяснено, что в исходных данных нет пропусков (Рис. 7). Следовательно, нужно найти выбросы в датасете. Была написана функция, которая для переданного признака подсчитывала количество выбросов (Рис. 16).

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рис. 16 – Функция нахождения выбросов.

После применения функции к признакам получились следующие результаты:

Изображение выглядит как текст, Шрифт, число, снимок экрана

Автоматически созданное описание

Рис. 17 – Выбросы.

Было решено предварительно ничего не делать с выбросами, если они не будут влиять на качество обучения модели. Далее мною был заменен тип всех переменных на int64 и была проведена нормализация данных (Рис. 18).

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание

Рис. 18 – Нормализация.

* + 1. **Разделение на тренировочную и тестовую выборки.**

Для разделения данных на тестовую и тренировочную выборки использовался train\_test\_split () библиотеки sklearn (Рис. 19). В качестве параметров были переданы X, y, test\_size и random\_state, где X – это преобразованный датасет без целевого признака, y – это значения целевого признака («Units Sold»), test\_size – говорит о том, в каком соотношении будут разделены данные на тренировочные и тестовые. В данном случае размер тренировочных данных равен 20 процентам. А параметр random\_state фиксирует случайное распределение выбора элементов из датасета, чтобы при повторном выполнении данные были такие же, как и при первом.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рис. 19 – Разделение данных.

* 1. **Обучение модели и анализ данных.** 
     1. **Построение класса модели нейронной сети.**

После разделения данных на тренировочные и тестовые был написан класс MLP нейронной сети. Данный класс представляет собой полносвязную нейронную сеть с 5 слоями. В нее включены 4 скрытых слоя и один входной. (Рис. 20). Первый слой принимает в себя 22 признака (22 столбца нашего датасета) и состоит из 128 нейронов. Второй слой состоит 64 нейронов. Третий слой состоит из 32 нейронов. Четвертый слой состоит из 16 нейронов, а выходной слой состоит из 1 нейрона. Так как это полносвязная сеть, то в ней все нейроны одного слоя связаны со всеми нейронами другого слоя. Чтобы привнести нелинейность вычислений, используется функция relu. Ее можно получить несколькими способами, в том числе и написать вручную, но был использован метод библиотеки pytorch.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 20 – Класс полносвязной нейронной сети.

* + 1. **Обучение модели на тренировочных данных.**

Следом, данная модель обучается на тренировочной выборке (Рис. 21). В данном коде происходит обучение многослойной перцептронной (MLP) модели для решения задачи регрессии. Сначала создается экземпляр MLP модели с помощью определения класса FullyConnected, который наследуется от nn.Module. Внутри класса, как уже было упомянуто, определяются 5 полносвязных слоев, включая выходной слой и четыре скрытых слоя. Затем определяются критерий ошибки (MSELoss) и оптимизатор (SGD) для обучения модели. Для обучения модели используются тренировочные данные, но параллельно идет вычисление для тестовой выборки. Это поможет определить переобучение модели в случае такового. Затем происходит цикл обучения модели, где на каждой итерации вычисляется значение функции потерь (MSE) для тренировочных и тестовых данных. Кроме того, вычисляются другие метрики качества, такие как коэффициент детерминации (R2), средняя абсолютная ошибка (MAE) и корень среднеквадратичной ошибки (RMSE), которые сохраняются в соответствующие массивы. Значения функции потерь для каждой эпохи выводятся на экран. По достижении порогового значения функции потерь (MSE\_LOSS > 200) обучение останавливается. Затем модель используется для предсказания значений целевого признака для тестовой и тренировочной выборок.

Изображение выглядит как текст, снимок экрана, документ

Автоматически созданное описание

Рис. 21 – применение модели.

* + 1. **Анализ результатов обучения.**

После обучения модели нужно проверить переобучена модель или нет. Переобучение достаточно легко определить. Если ошибка на тренировочной выборке падает, а на тестовой выборке растет, это значит, что модель слишком сильно приспособилась к тренировочным данным, другими словами, переобучилась. Чтобы была возможность наглядно сравнить ошибку на тестовой и тренировочной выборке, нужно добавить ее в два списка. На рисунке 21 представлены эти списки. Их названия mse\_history и mse\_history\_test для тренировочной и тестовой выборки соответственно. На каждой эпохе обучения модели данные списки расширяются значениями ошибки mse.

Итак, на рисунке 22 представлен график ошибки mse модели в зависимости от эпохи. На этом графике синяя линия – ошибка на тренировочной выборке, а оранжевая – на тестовой. Как видно из графика, ошибка на тестовой выборке всегда немного выше, чем на тренировочной. Однако ситуации, когда синяя линия идет вниз, а оранжевая вверх, не наблюдается. Также можно заметить, что взяты списки не целиком, а с четырехсотого элемента. Это сделано специально, чтобы лучше видеть динамику обучения на конечных эпохах. На рисунке 23 представлен график изменения mse на полных данных. Видно, что из-за первых примерно четырехсот эпох, слабо выражены на графике изменения на последних эпохах, так как значения первых эпох очень большие.

Изображение выглядит как текст, снимок экрана, линия, Параллельный

Автоматически созданное описание

Рис. 22 – График ошибки MSE

Изображение выглядит как текст, снимок экрана, линия, Параллельный

Автоматически созданное описание

Рис. 23 – График ошибки MSE целиком

Можно расмотреть еще несколько метрик, на сколько точно модель смогла предсказать значения для тестовых данных (Рис. 24). На данном рисунке представлены 4 метрики и количество эпох, которых потребовалось модели для обучения.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 24 – Метрики тестовой выборки.

На рисунке 25 представлены изменения метрик в зависимости от эпохи на тренировочной выборке. Можно заметить, что почти все метрики примерно к 500 эпохе кардинально снижаются и затем улучшаются заметно медленнее.

Изображение выглядит как диаграмма, снимок экрана, текст, линия

Автоматически созданное описание

Рис. 25 – Метрики на тренировочной выборке в зависимости от эпохи.

Теперь посмотрим на отличие между предсказанными значениями и реальными. Для этого можно использовать scatter и plot библиотеки sklearn. (Рис. 26, Рис. 27). Но на данных визуальных представлениях довольно сложно выделить и разглядеть насколько точно модель попадает при прогнозировании. По этой причине я выделил максимальный разброс между предсказанным значением и реальным на всей тестовой выборке (Рис. 28). Это значение составляет 9.63%.

Изображение выглядит как текст, снимок экрана, График, линия

Автоматически созданное описаниеИзображение выглядит как снимок экрана, текст, График, Шрифт

Автоматически созданное описание

Рис. 26 – Распределение значений. Рис. 27- График значений.

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание

Рис. 28 - Максимальная разница между предсказанным значением и реальным

Другим способом более подробно увидеть насколько точно модель предсказывает значения является уменьшение количества отображаемых точек данных на графиках. На рисунках 29 и 30 представлены данные для первых 150 значений.

Изображение выглядит как текст, снимок экрана, График, линия

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, Шрифт, График

Автоматически созданное описание

Рис. 29 Распределение значение (первые 150). Рис. 30 График значений (первые 150).

Из всего выше сказанного понятно, что данная модель неплохо предсказывает будущие значения. Но для полного понимания насколько хороша эта модель, было решено сравнить ее результаты с моделью на основе CNN, то есть сверточных нейронных сетей.

* + 1. **Построение CNN модели глубокого обучения.**

На рисунке 31 представлен код. Этот код определяет класс нейронной сети CNN, которая является сверточной нейронной сетью (Convolutional Neural Network). Эта модель используется для анализа и обработки данных в виде одномерных временных рядов. Класс CNN состоит из следующих слоев:

* Conv1d: первый сверточный слой с ядром (фильтром) размером 3 и смещением (padding) размером 1, который применяет 16 фильтров к входным данным с одним каналом (количество каналов определяет количество признаков входных данных).
* MaxPool1d: первый слой подвыборки с ядром размером 2, который уменьшает размерность выходных данных из предыдущего слоя вдвое.
* Conv1d: второй сверточный слой с ядром размером 3 и смещением размером 1, который применяет 32 фильтра к выходным данным предыдущего слоя.
* MaxPool1d: второй слой подвыборки с ядром размером 2, который уменьшает размерность выходных данных из предыдущего слоя вдвое.
* Linear: первый полносвязный слой с 64 скрытыми нейронами, который принимает на вход данные, полученные после операции вытягивания (flatten) из предыдущего слоя.
* Linear: второй полносвязный слой с 32 скрытыми нейронами.
* Linear: выходной слой с одним нейроном, который предсказывает значение целевой переменной.

Метод forward принимает входной тензор x и производит прямое распространение через сеть. В начале добавляется одна размерность к тензору x, чтобы получить тензор формы (batch\_size, 1, sequence\_length). Это необходимо, так как PyTorch ожидает, что входные тензоры для сверточных слоев будут иметь форму (batch\_size, num\_channels, sequence\_length), где num\_channels - количество каналов входного тензора (в нашем случае 1) и sequence\_length - длина последовательности. Затем x пропускается через первый сверточный слой с 16 фильтрами (ядро свертки размером 3 и паддингом 1) и применяется функция активации ReLU. Затем применяется операция макс-пулинга с ядром размером 2. Это уменьшает размер последовательности в два раза и помогает избежать переобучения. Далее повторяются аналогичные шаги со вторым сверточным слоем (уже с 32 фильтрами) и применяем макс-пулинг.

Затем преобразуется выход сверточного слоя в плоский вектор с помощью view. Далее этот вектор проходит через два полносвязных слоя с функцией активации ReLU и выходит на последний полносвязный слой, который содержит только один нейрон (поскольку наша задача - регрессия). Выход этого слоя представляет собой прогнозируемое значение для каждого элемента входной последовательности.

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рис. 31 – модель CNN

* + 1. **Обучение новой модели на тренировочных данных**

После этого было запущено обучение модели (Рис. 32). Данный код очень похож на тот, что использовался в предыдущей модели, но в нем есть отличия. Самое главное отличие – это использование специального метода из pytorch, который может динамически изменять learning\_rate оптимизатора в зависимости от ошибки на эпохе. Называется этот метод «ReduceLROnPlateau». Уже из названия понятно, что он делает. Когда наступает плато и модель не может найти минимум с текущим learning rate, она понижает его. То на сколько будет понижен learning rate определяется параметром factor. В данном случае понижается на половину. Параметр patience говорит о том, через какое количество похожих значений наступает плато. А параметр verbose, регулирует выводить сообщение об изменении learning rate или нет.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 32 – обучение CNN модели

* + 1. **Анализ результатов обучения новой модели.**

Стоит отметить, что несмотря на адаптивную регулировку learning rate и более продвинутую архитектуру сети, не удалось за вменяемое количество времени достичь значения целевой ошибки. Модели потребовалось более 7000 эпох чтобы приблизиться к mse = 300 (Рис. 33) при том, что целевое mse было 200. Из-за этого пришлось вручную остановить на 7142 эпохе с mse равным 307 на тренировочной выборке. Важно заметить, что модель не переобучилась.

Изображение выглядит как снимок экрана, линия, текст, График

Автоматически созданное описание

Рис. 33 – зависимость mse от эпох.

У данной модели следующие метрики на тестовой выборке:

Изображение выглядит как текст, Шрифт, белый

Автоматически созданное описание

Рис. 34 – Метрики на тестовой выборке.

В процессе обучения метрики тренировочной выборки менялись следующим образом:

Изображение выглядит как диаграмма, текст, линия, Параллельный

Автоматически созданное описание

Рис. 35 – Изменение метрик тренировочной выборки.

Так как метрика mse не достигла целевой, то понятно, что модель будет предсказывать хуже, чем предыдущая (Рис. 36).

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание

Рис. 36 – максимальное расхождение.

* + 1. **Построение 3 модели глубоко обучения.**

В качестве финальной проверки была построена 3-я модель. Она аналогично первой построена на mlp архитектуре. Но в отличие от первой, в ней всего 3 слоя и количество нейронов в этих слоях также меньше (Рис. 37).

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рис. 37 – mlp нейронная сеть

* + 1. **Обучение 3 модели на тренировочных данных**

Кроме этого, при обучении модели используется оптимизатор adam вместо sgd в первой модели (Рис. 38). Соответственно learning rate был поставлен другой В остальном сильных изменений при обучении нет.

Изображение выглядит как текст, снимок экрана, документ, Шрифт

Автоматически созданное описание

Рис. 38 – обучение 3 модели.

* + 1. **Анализ результатов обучения 3 модели.**

На рисунке 39 представлено изменение метрики mse на тренировочной и тестовой выборках в зависимости от эпохи. Видно, что количество эпох более 10000, но из-за легковесности самой модели, центральному процессору гораздо проще просчитать эти эпохи. Именно поэтому cnn модель считала 7000 эпох гораздо дольше, чем mlp 10000. Также видно, что переобучения не было.

Изображение выглядит как текст, снимок экрана, линия, График

Автоматически созданное описание

Рис. 39 – зависимость MSE от эпохи.

Основные метрики для 3 модели:

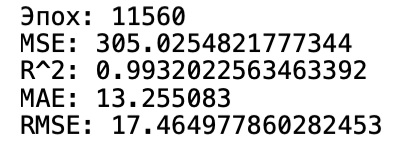


Рис. 40 – основные метрики.

Изменения метрик в зависимости от эпохи на тренировочной выборке (Рис. 41).

Изображение выглядит как диаграмма, текст, линия, Параллельный

Автоматически созданное описание

Рис. 41 - Метрики на тренировочной выборке в зависимости от эпохи.

Распределение реальных и предсказанных значений (Рис. 42). График реальных значений и предсказанных, наложенных друг на друга (Рис. 43).

Изображение выглядит как текст, снимок экрана, линия, График

Автоматически созданное описаниеИзображение выглядит как снимок экрана, График

Автоматически созданное описание

Рис. 42 – распределение значений. Рис. 43 – график значений.

Но как уже было упомянуто выше, на этих визуализациях не очень понятно, насколько точно модель предсказывает. Поэтому возьмем также первые 150 значений (Рис. 44, Рис. 45)

Изображение выглядит как текст, снимок экрана, График, линия

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, График, диаграмма

Автоматически созданное описание

Рис. 44 – распределение значений (150) Рис. 45 – график значений (150)

Заметно, что модель очень неплохо предсказывает значения тестовой выборки. Поэтому нужно посмотреть какое максимальное расхождение между предсказанными и действительными значениями (Рис. 46). Расхождение составило даже меньше, чем в первой модели. То есть данная модель, ровно как и первая, очень неплохо справилась со своей задачей.

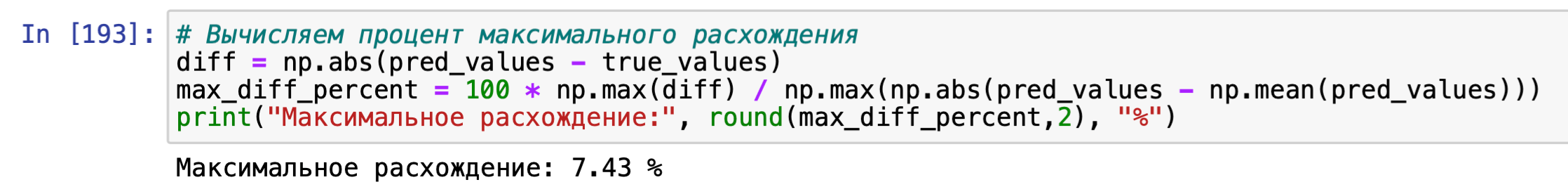


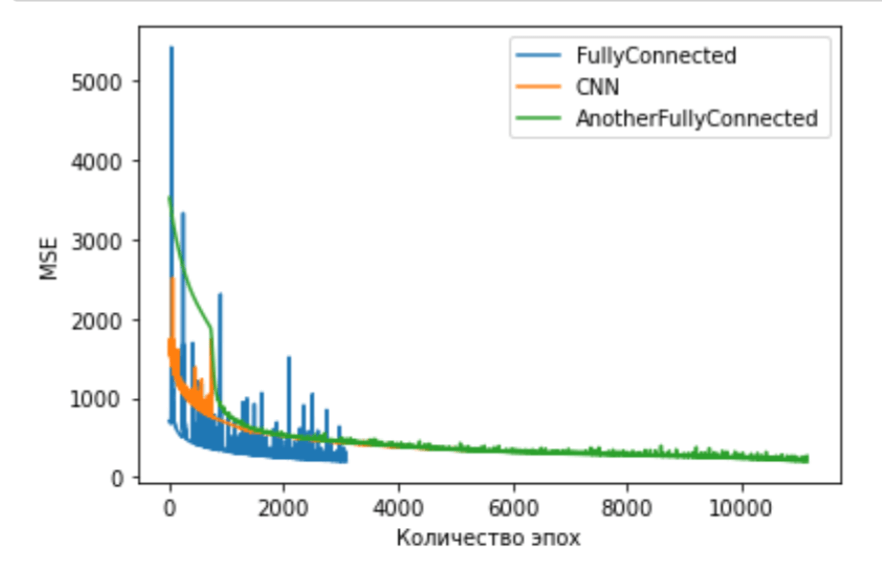
Рис. 46 – Максимальное расхождение.

1. **Выводы.**
   1. **Анализ результатов работы.**

Выведем метрики mse всех моделей отдельно для тренировочной выборки и тестовой выборки. (Рис. 47, Рис. 48). Из этих данных можно выделить, что все 3 модели смогли адаптироваться к данным и делать более-менее релевантные прогнозы. Так как с моей точки зрения, ошибка примерно в 10 процентов является приемлемой для прогнозирования конкретно этих данных.

Невозможно не выделить CNN модель. Несмотря на то, что она смогла адаптироваться к данным и показать достойный результат, она обучалась гораздо медленнее MLP моделей. Если смотреть по эпохам, то за одинаковое количество эпох ошибка падает примерно одинаково, но CNN модель проходит одну эпоху гораздо дольше по времени чем остальные модели. Плюсом же CNN является то, что в процессе обучения ошибка почти не скачет и планомерно уменьшается, чего не скажешь о MLP моделях. Даже в 3 модели, где используется более продвинутый оптимизатор adam вместо sgd не наблюдается такого стабильного понижения ошибки.

Что касается MLP моделей, то они показали себя очень хорошо. Модель в которой 3 слоя в конечном итоге показала себя лучше по максимальному расхождению между действительными и предсказанными значениями при одинаковой mse ошибке на тренировочной выборке, но ей потребовалось на это больше эпох. При этом по времени обучения модели оказались примерно равны. Поэтому, я считаю 3 модель лучше всех остальных. Она самая простая по структуре, не требует больших вычислительных мощностей и очень хорошо предсказывает значения.

Изображение выглядит как текст, снимок экрана, диаграмма, линия

Автоматически созданное описание

Рис. 47 – тренировочная выборка. Рис. 48 – тестовая выборка.

* 1. **Общие выводы по использованию методов глубокого обучения для прогнозирования объема потребления**

Подводя итоги, можно сказать, что глубокое обучение неплохо подходит для прогнозирования объема потребления. Также можно точно сказать, что для конкретно моего датасета лучше всего использовать MLP (полносвязные) модели. Они лучше справляются с задачей регрессии в моей курсовой работе чем сверточная сеть. Что касается структуры таких сетей, то определить какая лучше – это уже более сложный вопрос. В моем случае немного лучше себя показала более простая трехслойная модель.

**Список использованных источников**

Основные источники:

1. <https://www.kaggle.com/datasets/heemalichaudhari/adidas-sales-dataset>
2. <https://pytorch.org/docs/stable/index.html>
3. <https://scikit-learn.org/stable/index.html>
4. <https://pandas.pydata.org>
5. М.В. Коротеев. Учебное пособие по дисциплине “Анализ данных и машинное обучение” - 2018.
6. Себастьян Рашка “Python Machine Learning ” - 2015.

Дополнительные источники:

1. Eli Stevens, Luca Antiga, Thomas Viehmann “Deep Learning with PyTorch” - 2020.
2. Sherin Thomas и Sudhanshu Passi “PyTorch Deep Learning Hands-On”.
3. Петерр Флах. “Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных” – 2015.