ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ БЮДЖЕТНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ»**

**(ФИНАНСОВЫЙ УНИВЕРСИТЕТ)**

Департамент анализа данных и машинного обучения

Пояснительная записка к курсовой работе по дисциплине

«Технологии анализа данных и машинного обучения»

на тему:

**«Прогнозирование и анализ выручки предприятия с помощью нейронных сетей»**

*Выполнил:*

студент группы ПИ20-3

Федоров Александр Константинович

*Научный руководитель:*

доцент д.ф-м.н. Золотова Татьяна Валерьяновна.

**Москва 2023**

Оглавление

[Введение 3](#_Toc134653604)

[Глава 1. Теоретическая часть 5](#_Toc134653605)

[1.1. Определение основных понятий и терминов 5](#_Toc134653606)

[1.2. Преимущества и недостатки нейронных сетей в прогнозировании и анализе выручки предприятия 6](#_Toc134653607)

[1.3. Методы прогнозирования и анализа с помощью нейронных сетей 7](#_Toc134653608)

[Глава 2. Практическая часть 9](#_Toc134653609)

[2.1. Загрузка, очистка и описание данных 9](#_Toc134653610)

[2.2 Разделение данных на обучающую, тестовую и валидационную выборки 13](#_Toc134653611)

[2.3 Создание моделей, их обучение и анализ 16](#_Toc134653612)

[2.3.1 Построение, обучение и анализ 1 модели 16](#_Toc134653613)

[2.3.2 Построение, обучение и анализ 2 модели 20](#_Toc134653614)

[2.3.3 Построение, обучение и анализ 3 модели 23](#_Toc134653615)

[2.3.4 Построение, обучение и анализ 4 модели 26](#_Toc134653616)

[2.4 Сравнение моделей 30](#_Toc134653617)

[Глава 3. Заключение 31](#_Toc134653618)

[Список использованных источников 32](#_Toc134653619)

# **Введение**

Выручка является одним из ключевых показателей финансовой деятельности предприятия, она отражает объем доходов, полученных за определенный период времени. Роль выручки в бизнесе трудно переоценить - она является главным источником финансирования различных проектов и направлений деятельности компаний. Эффективный анализ и прогнозирование выручки позволяет бизнесу принимать обоснованные решения и оптимизировать свою деятельность.

Прогнозирование и анализ выручки являются необходимыми задачами для любого предприятия, которое стремится к успеху и процветанию. Они позволяют предприятию оценить реальную ситуацию на рынке, спрогнозировать будущие доходы и принимать обоснованные решения на основе этих прогнозов.

Без эффективного прогнозирования и анализа выручки предприятие может столкнуться с множеством проблем, таких как недостаток финансовых ресурсов для инвестирования в развитие бизнеса, неправильное планирование закупок и производства, недостаточное количество продукции для удовлетворения спроса и многие другие.

Примерами компаний, которые достигли успеха благодаря эффективному анализу и прогнозированию выручки, можно привести Amazon, Google, Apple и другие. Они не только активно используют новые технологии, включая нейронные сети, для прогнозирования выручки, но и активно разрабатывают их, чтобы улучшить свои бизнес-показатели и увеличить прибыль.

В связи с этим, прогнозирование и анализ выручки предприятия с помощью нейронных сетей является важной задачей в настоящее время. Именно поэтому основной целью работы является разработка моделей нейронных сетей, позволяющей прогнозировать будущую выручку и производить ее анализ.

Задачи данной работы:

* Изучить методы прогнозирования и анализа выручки предприятия с помощью нейронных сетей.
* Сбор и анализ данных о выручке предприятия
* Разработка и обучение нейронной сети для прогнозирования выручки предприятия
* Оценить точность модели и проанализировать результаты

# **Глава 1. Теоретическая часть**

## **1.1. Определение основных понятий и терминов**

В данной работе используются следующие основные понятия и термины:

Выручка – денежный объем, полученный предприятием за реализацию продукции, выполнение работ, оказание услуг в определенный период времени.

Прогнозирование – процесс определения возможного будущего состояния объекта или явления на основе известных данных и использования математических методов и моделей.

Нейронные сети – это компьютерные системы, которые имитируют функцию работы человеческого мозга. Они состоят из множества связанных между собой узлов (нейронов), которые передают и обрабатывают информацию. Каждый нейрон принимает входные данные, обрабатывает их и передает результат следующему нейрону в сети. Одной из основных задач нейронных сетей является обучение на основе большого количества данных, что позволяет им находить закономерности и решать сложные задачи, такие как классификация, регрессия, обработка естественного языка и другие.

Функция активации – это нелинейная математическая функция, которая определяет выходное значение нейрона на основе его входного сигнала. Она используется для введения нелинейности в нейронные сети и позволяет им моделировать сложные нелинейные зависимости между входными и выходными данными.

Существует множество функций активации, каждая из которых подходит для разных задач и имеет свои преимущества и недостатки. Некоторые из наиболее распространенных функций активации включают в себя: сигмоидальная функция, гиперболический тангенс, ReLU, Softmax.

Скрытый слой в нейронной сети представляет собой набор нейронов, которые не связаны напрямую с входом или выходом сети. Эти нейроны обрабатывают входные данные и передают их на следующий слой, который может быть скрытым или выходным. Скрытый слой является ключевым элементом нейронной сети, так как именно он позволяет сети извлекать высокоуровневые признаки из исходных данных и обрабатывать их для решения задачи. Количество скрытых слоев и количество нейронов в каждом слое может варьироваться в зависимости от задачи и размера входных данных. Оптимальное количество слоев и нейронов может быть найдено путем экспериментов и оптимизации параметров сети.

Функция потерь — это функция, которая используется в машинном обучении для измерения ошибки модели на заданном наборе данных. Она описывает, насколько хорошо модель справляется с задачей, которую ей необходимо решить. Чем меньше значение функции потерь, тем лучше модель работает. Примеры функций потерь: среднеквадратичная ошибка (MSE), категориальная перекрестная энтропия (Categorical Cross-Entropy), средняя абсолютная ошибка (MAE).

Обратное распространение ошибки – это метод обучения нейронных сетей, позволяющий оптимизировать веса между нейронами, чтобы минимизировать ошибку на выходном слое.

Метрики качества – это инструменты для оценки производительности нейронной сети. Они используются для измерения того, насколько хорошо модель выполняет задачу, для которой она обучалась. Некоторые из распространенных метрик качества: точность (accuracy), среднеквадратичная ошибка (MSE), Полнота (recall), F1-мера, Коэффициент детерминации (R2).

## **1.2. Преимущества и недостатки нейронных сетей в прогнозировании и анализе выручки предприятия**

Нейронные сети имеют ряд преимуществ в прогнозировании и анализе выручки предприятия:

* Обработка большого объема данных.
* Выявление сложных связей.
* Прогнозирование трендов и изменений.
* Автоматизация процесса прогнозирования.
* Гибкость.
* Улучшенная точность.

Несмотря на многочисленные преимущества, нейронные сети также имеют некоторые недостатки в прогнозировании и анализе выручки предприятия:

Необходимость большого количества данных для достижения высокой точности.

* Сложность интерпретации результатов.
* Возможность переобучения модели.
* Высокая вычислительная сложность.
* Необходимость специалистов в области машинного обучения для разработки и настройки моделей.
* Риски конфиденциальности данных при использовании сторонних сервисов обработки данных.

## **1.3. Методы прогнозирования и анализа с помощью нейронных сетей**

Методы прогнозирования и анализа с помощью нейронных сетей включают в себя много различных алгоритмов и подходов. Некоторые из них:

1. Многослойные перцептроны (MLP) - это наиболее распространенный тип нейронных сетей для прогнозирования и анализа данных. Они состоят из нескольких слоев нейронов, каждый из которых связан со следующим слоем. MLP может быть использован для решения задач классификации и регрессии.
2. Рекуррентные нейронные сети (RNN) - это тип нейронных сетей, способных обрабатывать последовательности данных. Они имеют связи обратной связи, что позволяет использовать предыдущие выходы для обработки последующих входов. RNN можно использовать для решения задач, таких как прогнозирование временных рядов и обработка естественного языка.
3. Сверточные нейронные сети (CNN) - это тип нейронных сетей, которые используются для обработки изображений и других типов двухмерных данных. Они используют сверточные слои для выделения признаков из входных данных.
4. Глубокие нейронные сети (DNN) - это нейронные сети, состоящие из более чем одного скрытого слоя. Они могут использоваться для решения задач, которые требуют более сложных вычислений.
5. Автокодировщики - это нейронные сети, используемые для сжатия и восстановления данных. Они могут быть использованы для анализа и визуализации данных.

Кроме того, есть множество других методов, таких как генеративно-состязательные сети (GAN), длинные краткосрочные памяти (LSTM) и другие. Каждый метод имеет свои особенности и применяется в зависимости от задачи, которую необходимо решить.

В данной работе я буду использовать многослойный перцептрон(MLP), сверточные нейронные сети (CNN) и глубокие нейронные сети (DNN).

# **Глава 2. Практическая часть**

## **2.1. Загрузка, очистка и описание данных**

Чтобы начать работу с данными, необходимо их загрузить в рабочую среду. Для этого можно использовать различные методы, в зависимости от формата данных и используемой программной среды.

Для загрузки данных в Jupyter Notebook с помощью библиотеки Pandas можно использовать метод read\_csv()(рисунок 1).

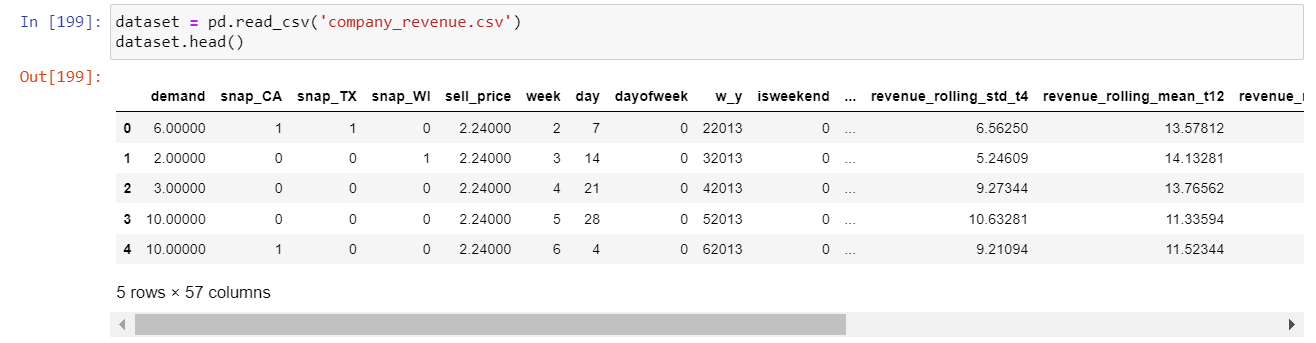


Рисунок 1 – Загрузка данных в Jupyter Notebook

После загрузки нужно убрать не нужные столбцы (рисунок 2).

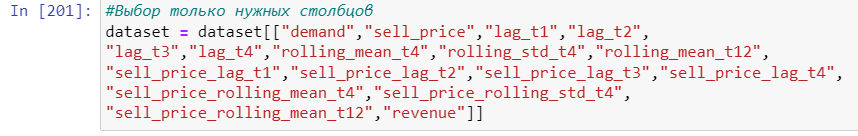


Рисунок 2 – Выбор нужных столбцов

В данном датасете используются 16 признаков:

* demand - спрос на товары
* sell\_price - средняя цена продажи товаров
* lag\_t1 - спрос на товаров в предыдущий момент времени (t-1)
* lag\_t2 - спрос на товаров в момент времени еще более ранний, чем lag\_t1 (t-2)
* lag\_t3 - спрос на товаров в момент времени еще более ранний, чем lag\_t2 (t-3)
* lag\_t4 - спрос на товаров в момент времени еще более ранний, чем lag\_t3 (t-4)
* rolling\_mean\_t4 - скользящее среднее значение спроса на товары за последние 4 момента времени
* rolling\_std\_t4 - скользящее стандартное отклонение спроса на товары за последние 4 момента времени
* rolling\_mean\_t12 - скользящее среднее значение спроса на товары за последние 12 моментов времени
* sell\_price\_lag\_t1 - средняя цена продажи товаров в предыдущий момент времени (t-1)
* sell\_price\_lag\_t2 - средняя цена продажи товаров в момент времени еще более ранний, чем sell\_price\_lag\_t1 (t-2)
* sell\_price\_lag\_t3 - средняя цена продажи товаров в момент времени еще более ранний, чем sell\_price\_lag\_t2 (t-3)
* sell\_price\_lag\_t4 - средняя цена продажи товаров в момент времени еще более ранний, чем sell\_price\_lag\_t3 (t-4)
* sell\_price\_rolling\_mean\_t4 - скользящее среднее значение цены продажи товаров за последние 4 момента времени
* sell\_price\_rolling\_std\_t4 - скользящее стандартное отклонение цены продажи товаров за последние 4 момента времени
* sell\_price\_rolling\_mean\_t12 - скользящее среднее значение цены продажи товаров за последние 12 моментов времени
* revenue – выручка

После загрузки данных, необходимо провести их анализ и описание для дальнейшего использования в модели. Для этого необходимо рассмотреть основные характеристики датасета, такие как количество строк и столбцов, типы данных, наличие пропущенных значений и выбросов, а также статистические показатели.

Вначале рассмотрю количество строк и столбцов. Количество строк и столбцов в датасете можно узнать с помощью метода shape в pandas. В моем датасете 17 столбцов и 100000 строк (рисунок 3)

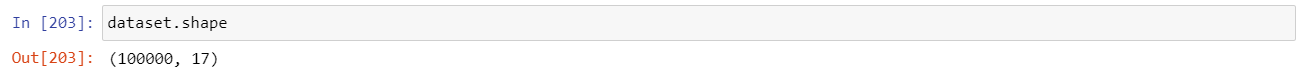


Рисунок 3 – Количество строк и столбцов датасета

После получения информации о количестве строк и столбцов, следующим шагом может быть использование метода info для получения более подробной информации о датасете, включая типы данных, наличие пропущенных значений и общее количество записей в каждом столбце (рисунок 4).

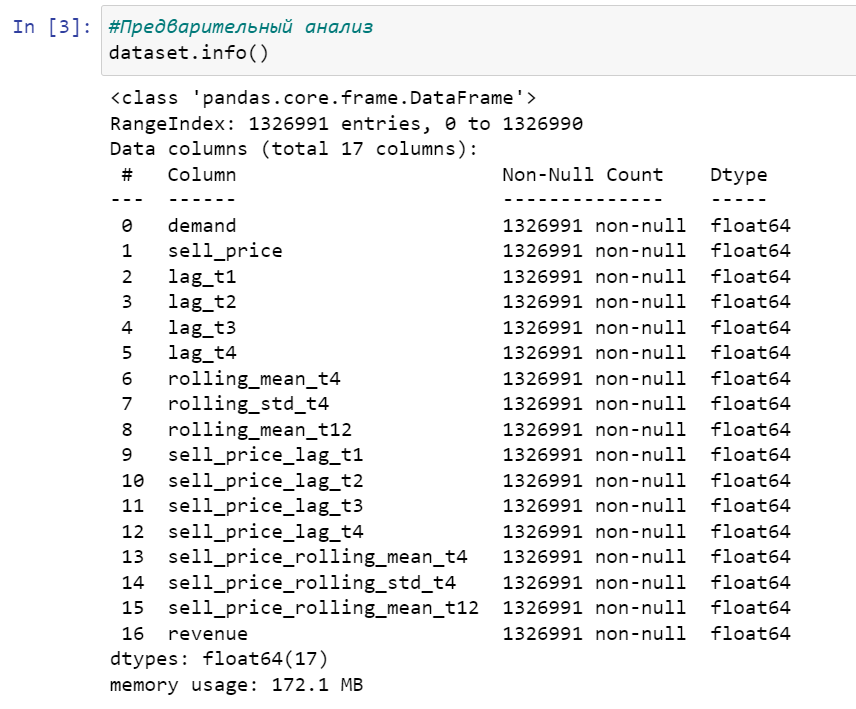


Рисунок 4 – Подробная информация о дата сете

Чтобы вывести информацию об отсутствующих значениях в датасете, можно воспользоваться методом isnull и sum, который позволяет подсчитать количество отсутствующих значений в каждом столбце (рисунок 5). Выяснилось, что в выбранных данных нет отсутствующих значений.

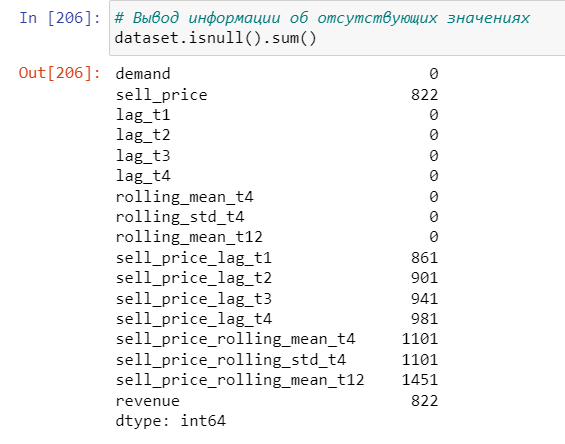


Рисунок 5 – Информация об отсутствующих значениях

Так как есть пустые значения нужно удалить такие строки используя метод dropna (рисунок 6).

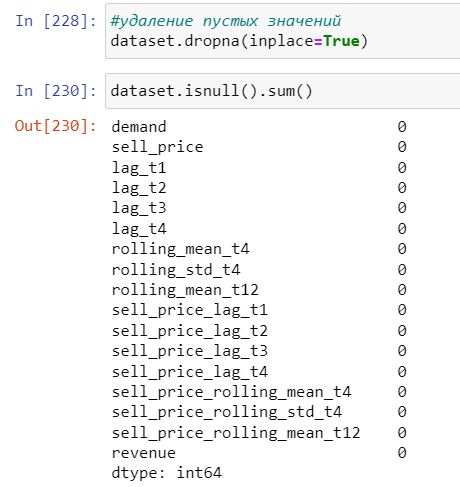


Рисунок 6 - Удаление пустых значений

Для получения основных статистических характеристик количественных данных в датасете использую метод describe (рисунок 7). Он возвращает объект DataFrame, который содержит следующую информацию:

* count - количество значений в столбце, исключая значения NaN (пропущенные значения)
* mean - среднее значение столбца
* std - стандартное отклонение столбца
* min - минимальное значение в столбце
* 25% - значение квартиля 25%
* 50% - значение квартиля 50% (медиана)
* 75% - значение квартиля 75%
* max - максимальное значение в столбце

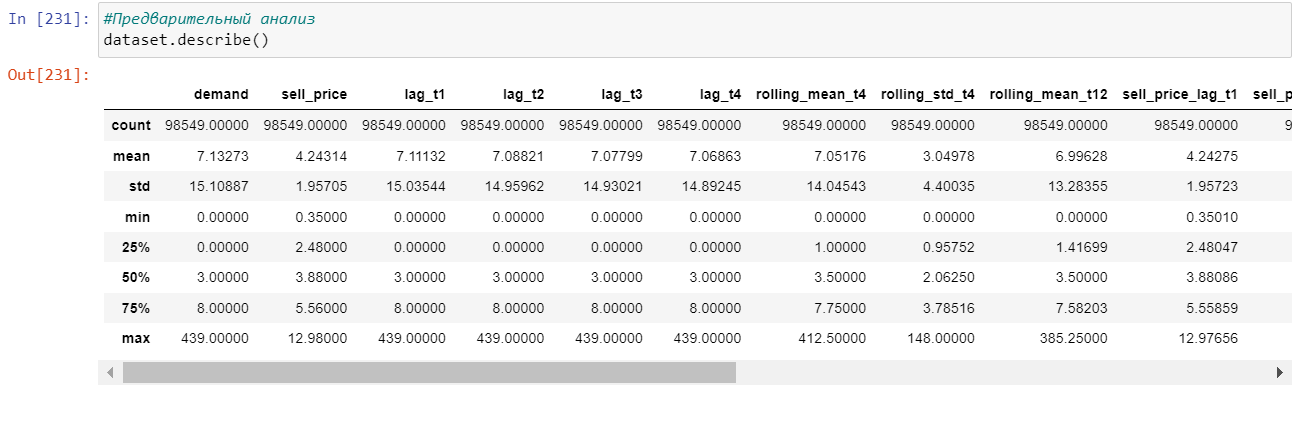


Рисунок 7 –основные статистических характеристики количественных данных датасета

## **2.2 Разделение данных на обучающую, тестовую и валидационную выборки**

Разделение данных на обучающую, тестовую и валидационную выборки является важным шагом в глубоком обучении нейронных сетей. Это позволяет оценить качество модели на неизвестных данных и выявить проблемы, связанные с переобучением.

Обучающая выборка используется для обучения модели, тестовая выборка - для оценки качества модели после ее обучения, а валидационная выборка - для настройки гиперпараметров модели и предотвращения переобучения (когда модель "запоминает" обучающую выборку, но плохо обобщает на новые данные).

Перед тем как разделять данные я создал класс MyDataset (рисунок 8), который является кастомным датасетом. Он позволяет определить, как данные будут загружаться, обрабатываться и подаваться в модель нейронной сети. Класс MyDataset имеет три метода:

* \_\_init\_\_: инициализирует экземпляр класса, принимая два аргумента: data\_X и data\_y, которые являются данными для X и y соответственно.
* \_\_getitem\_\_: возвращает элементы датасета с индексом idx. В данном случае, он возвращает x и y.
* \_\_len\_\_: возвращает количество элементов в датасете.

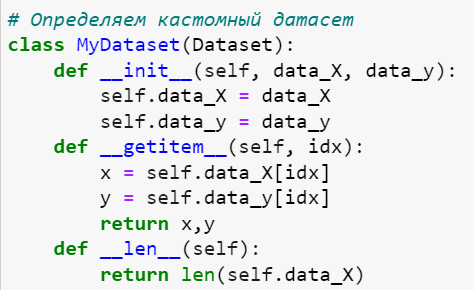


Рисунок 8 – Кастомный дата сет

На рисунке 9 показан процесс разделения набора данных на обучающую, тестовую и валидационную выборки, а затем преобразование каждой выборки в тензоры PyTorch. Кроме того, создаются кастомные датасеты и загружаются с помощью DataLoader.

Первым шагом в коде является разделение набора данных на обучающую, тестовую и валидационную выборки. Это важный шаг в процессе обучения нейронной сети, так как он позволяет измерить производительность модели на данных, которые ей не известны, и определить, насколько хорошо модель справляется с обобщением данных.

Для выполнения этого шага используется функция train\_test\_split из библиотеки scikit-learn, которая разделяет набор данных на две части: обучающую и тестовую выборки. Параметр test\_size=0.2 определяет, что 20% данных будут использоваться для тестирования модели. Затем снова используется функция train\_test\_split, но уже для разделения обучающей выборки на обучающую и валидационную выборки в соотношении 80/20 соответственно.

Затем каждая выборка преобразуется в тензоры PyTorch, которые являются основным типом данных в PyTorch и используются для обработки данных и вычисления градиентов в процессе обучения моделей глубокого обучения. Для этого используется функция torch.tensor, которая принимает в качестве аргумента разделенные данные при помощи функции train\_test\_split и возвращает тензор PyTorch. Метод float() используется для преобразования тензора в числа с плавающей запятой, что необходимо для обучения нейронных сетей.

Затем для каждой выборки создается кастомный датасет MyDataset, который определяет, как данные будут доступны во время обучения.

Для загрузки данных в модель используется DataLoader, который разбивает данные на батчи заданного размера. В нашем случае, для обучающей выборки мы используем DataLoader с параметром shuffle=True, что означает, что данные будут перемешиваться перед каждой эпохой обучения. Для тестовой и валидационной выборок shuffle=False, чтобы сохранить порядок данных и можно было посчитать метрики качества.

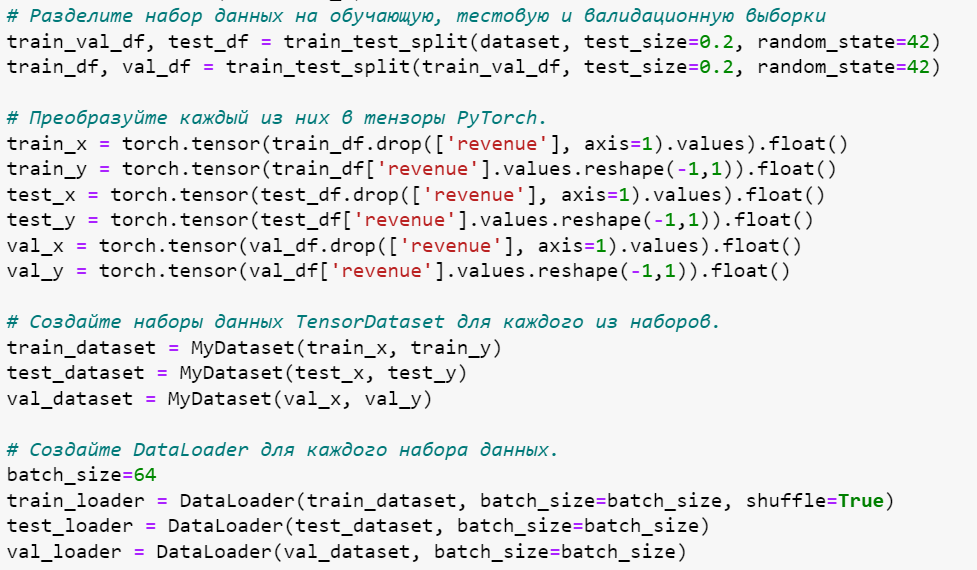


Рисунок 9 – Разделение данных на обучающую, тестовую и валидационную выборки

## **2.3 Создание моделей, их обучение и анализ**

### 2.3.1 Построение, обучение и анализ 1 модели

Данная модель представляет собой простую многослойную перцептронную нейронную сеть, состоящую из трех линейных слоев и функции активации ReLU(рисунок 10).

Архитектура модели:

* Входной слой размерности 16, соответствующей количеству признаков во входном векторе
* Скрытый слой размерности 32, с функцией активации ReLU
* Скрытый слой размерности 16, с функцией активации ReLU
* Выходной слой размерности 1, для регрессии.

Функция forward определяет прямой проход (forward pass) через нейронную сеть. На вход функция принимает входной тензор x, который проходит через линейный слой fc1, затем функцию активации ReLU, затем скрытый слой fc2 и еще одну функцию активации ReLU, и, наконец, выходной слой fc3. Результат возвращается из функции forward в виде тензора, содержащего предсказанные значения.

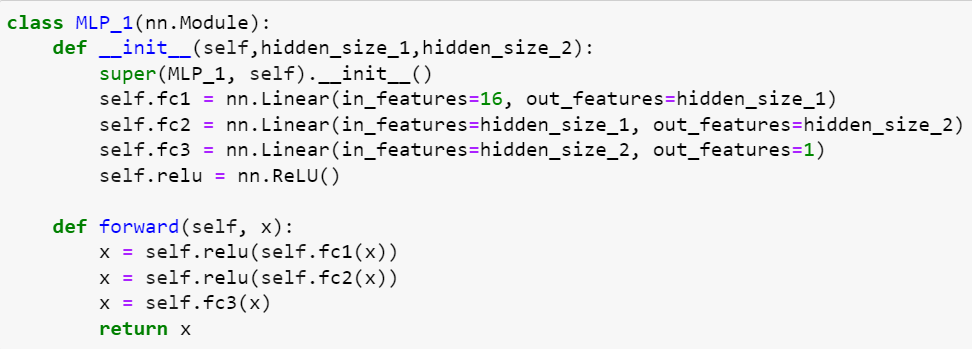


Рисунок 10 – Первая модель

Также определены функция потерь, которая используется для вычисления ошибки между предсказанными и истинными значениями, оптимизатор, который обновляет веса модели на каждой эпохе с целью минимизации функции потерь, и количество эпох, в течение которых будет обучаться модель. В первой модели функция потерь – MSELoss, а оптимизатор – Adam (рисунок 11). В скобках оптимизатора находится его параматры:

* model\_1.parameters() - передается список параметров модели, которые будут оптимизироваться (веса и смещения каждого слоя).
* lr=0.001 - learning rate (скорость обучения), коэффициент, определяющий величину изменения весов при обратном распространении ошибки.
* weight\_decay=0.01 - коэффициент регуляризации L2, который помогает бороться с переобучением модели. Он добавляет штраф к весам модели, чтобы уменьшить их абсолютные значения.

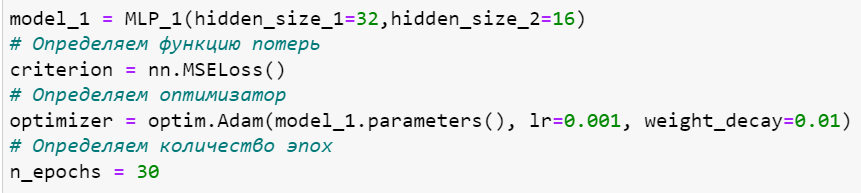


Рисунок 11 – Функция потерь, оптимизатор первой модели

После рассмотрения архитектуры модели обучим ее (рисунок 12). Сначала инициализируем два списка sp\_running\_loss и sp\_val\_loss, в которые будут сохраняться значения функции потерь на каждой эпохе для тренировочного и валидационного наборов соответственно.

Затем запускается цикл по эпохам. Количество эпох определили на рисунке 9. Внутри цикла мы инициализируем переменную running\_loss для хранения функции потерь на каждой итерации по тренировочному набору данных, а затем перебираем элементы в train\_loader (итератор для тренировочного набора данных). На каждой итерации мы получаем входные данные inputs и метки labels, обнуляем градиенты в оптимизаторе optimizer.zero\_grad(), делаем прямой проход через модель outputs = model\_1(inputs), вычисляем функцию потерь loss = criterion(outputs, labels), проводим обратный проход и обновляем веса с помощью оптимизатора optimizer.step(). Затем мы добавляем значение loss.item() к running\_loss для статистики обучения.

После прохождения всех итераций по тренировочному набору данных мы вычисляем функцию потерь на валидационном наборе данных аналогичным образом, сохраняем значения в списки sp\_running\_loss и sp\_val\_loss и выводим на экран значение функции потерь для тренировочного и валидационного наборов на текущей эпохе.

После окончания всех эпох мы проводим тестирование модели на тестовом наборе данных, используя test\_loader. Затем мы выводим на экран значение функции потерь для тестового набора данных и сообщаем, что обучение закончено.



Рисунок 12 – обучение модели

После обучения модели нужно провести анализ результатов обучения (рисунок 13). Для оценки качества модели я использовал такие метрики качества как: R2, MSE, MAE. В целом, значения метрик R2, MSE и MAE указывают на среднее качество работы модели

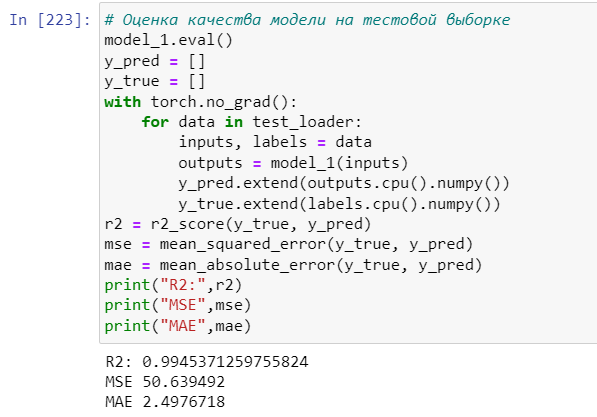


Рисунок 13 - Оценка качества модели

Чтобы посмотреть, как обучалась модель по эпохам на обучающей и валидационной выборке я построил график, который отображает изменение значения функции потерь (loss) в процессе обучения модели на обеих моделях (рисунок 14). Видно, что потери обучения и проверки уменьшаются на протяжении первых нескольких эпох, а затем начинают колебаться и изменяться нерегулярно. Это может означать, что модель достигла определенного уровня производительности и больше не может улучшаться

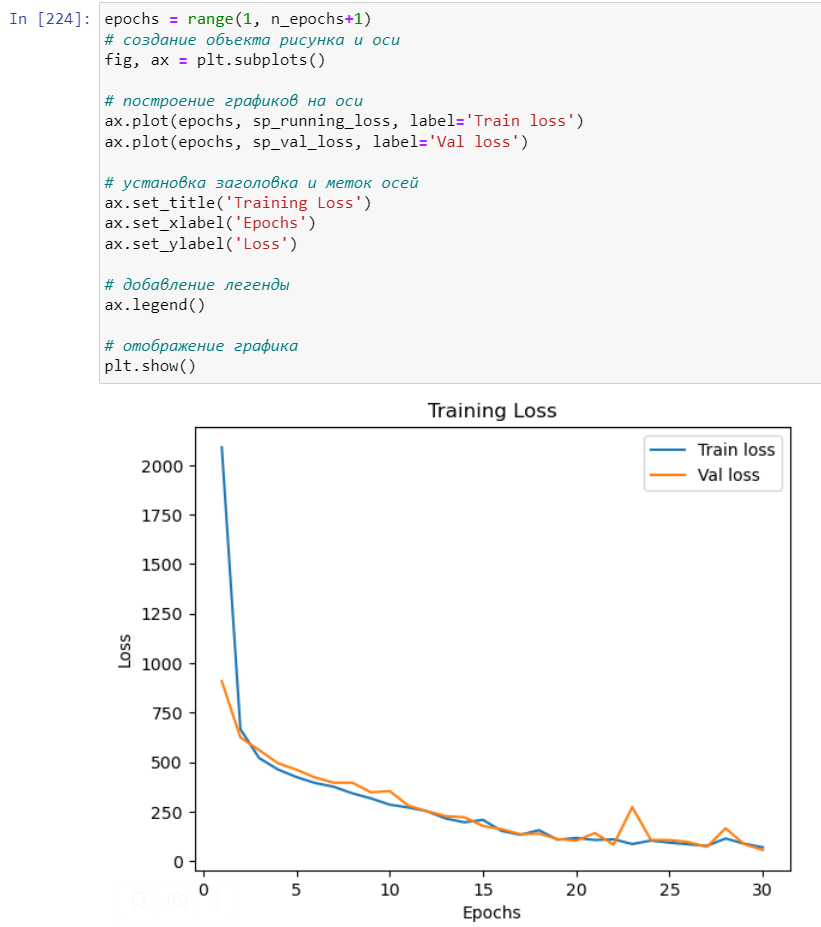


Рисунок 14 – График изменения значения функции потерь

### 2.3.2 Построение, обучение и анализ 2 модели

Данная модель представляет собой класс нейронной сети, основанной на MLP (рисунок 15).

Архитектура данной сети состоит из пяти слоев:

* Первый слой fc1 принимает на вход тензор размерности (batch\_size, 16) (16 - размерность входных данных) и преобразует его в тензор размерности (batch\_size, 32) (32 - размерность скрытого слоя).
* Второй слой fc2 принимает на вход тензор размерности (batch\_size, 32) и преобразует его в тензор размерности (batch\_size, 64).
* Третий слой fc3 принимает на вход тензор размерности (batch\_size, 64) и преобразует его в тензор размерности (batch\_size, 32).
* Четвертый слой fc4 принимает на вход тензор размерности (batch\_size, 32) и преобразует его в тензор размерности (batch\_size, 16).
* Пятый и последний слой fc5 принимает на вход тензор размерности (batch\_size, 16) и преобразует его в тензор размерности (batch\_size, 1) - это выходные данные.

Каждый слой в данной сети является полносвязным и имеет веса, которые будут обновляться во время обучения сети. Для активации выхода каждого слоя используется функция ReLU.

Метод forward принимает на вход тензор входных данных x и последовательно прогоняет его через каждый из пяти слоев, применяя к выходу каждого слоя функцию активации ReLU, кроме последнего слоя. Выход последнего слоя представляет собой предсказание модели и возвращается в качестве результата работы метода.

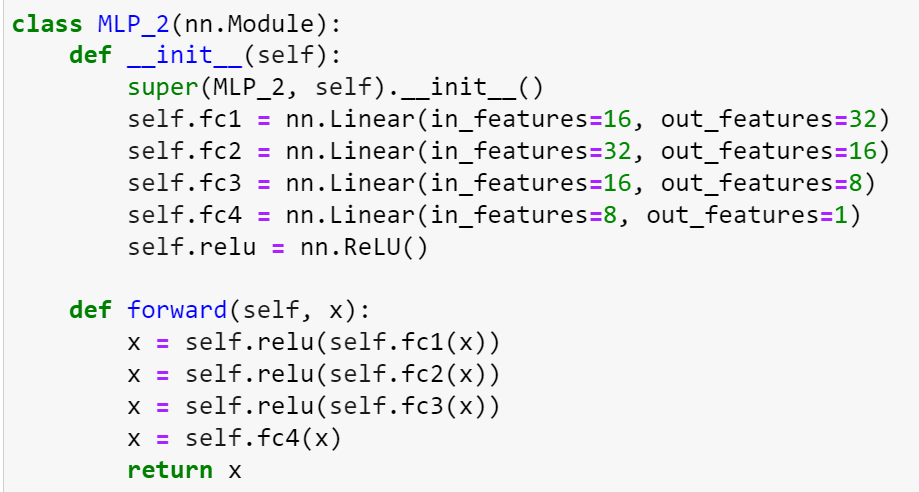


Рисунок 15 – Вторая модель

Также определены функция потерь, которая используется для вычисления ошибки между предсказанными и истинными значениями, оптимизатор, который обновляет веса модели на каждой эпохе с целью минимизации функции потерь, и количество эпох, в течение которых будет обучаться модель. Во второй модели функция потерь – MSELoss, а оптимизатор – Adam (рисунок 16).

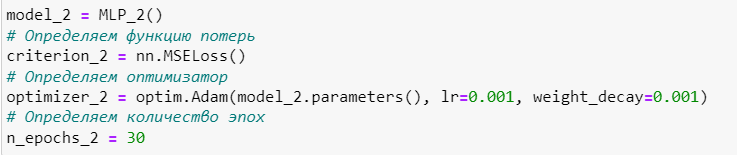


Рисунок 16 - Функция потерь, оптимизатор второй модели

Обучение происходит 2 модели происходит аналогично 1 модели (рисунок 17).



Рисунок 17 – Обучение второй модели

После обучения модели нужно провести анализ результатов обучения (рисунок 18). Для оценки качества второй модели я использовал такие метрики качества как: R2, MSE, MAE. В целом, значения метрик R2, MSE и MAE указывают на высокое качество работы модели. Ошибки второй модели меньше, чем у первой. Это значит, что вторая модель обучилась лучше.

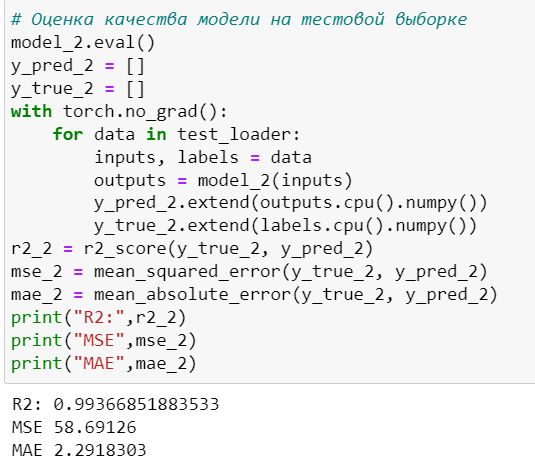


Рисунок 18 – Оценка качества второй модели

Чтобы посмотреть, как обучалась модель по эпохам на обучающей и валидационной выборке я построил график, который отображает изменение значения функции потерь (loss) в процессе обучения модели (рисунок 19). Видно, что потери обучения и проверки уменьшаются на протяжении всех эпох, но под конец затем начинают колебаться. Это означает, что модель достигла определенного уровня производительности.

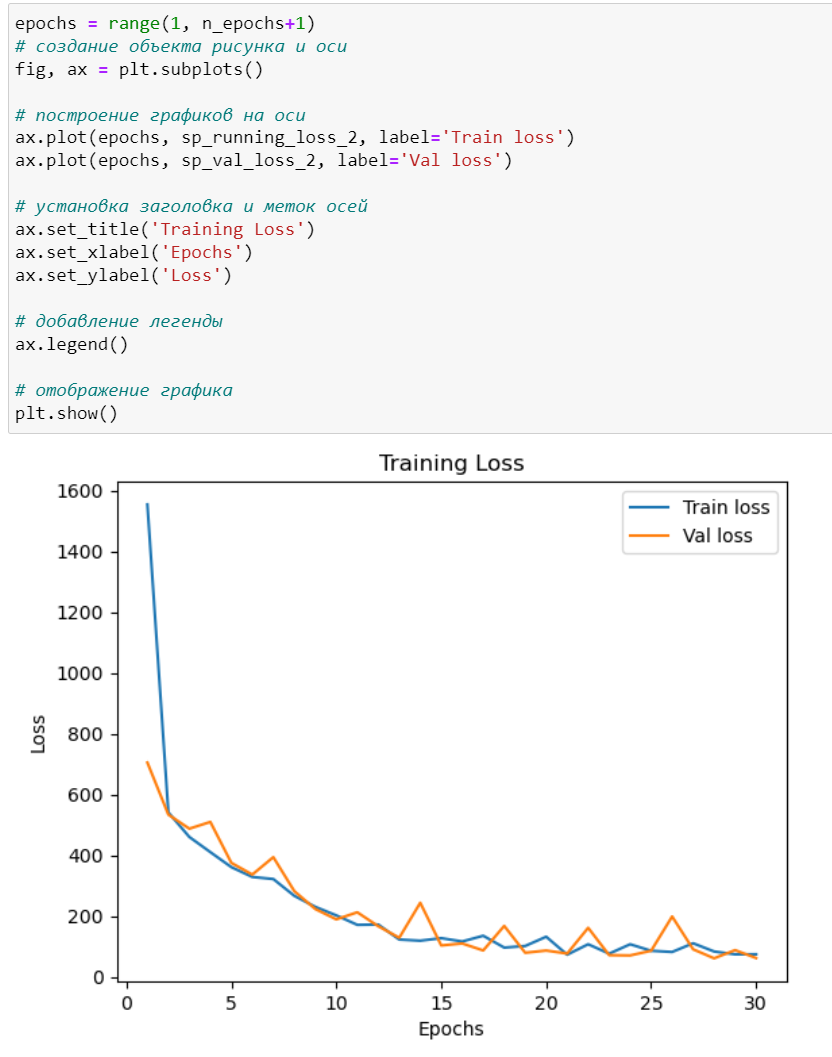


Рисунок 19 - График изменения значения функции потерь второй модели

### 2.3.3 Построение, обучение и анализ 3 модели

Данная модель (рисунок 20) представляет собой сверточную нейронную сеть(CNN).

Архитектура данной модели состоит из следующих слоев:

* Слой свертки Conv1d с ядром размера 3, количеством входных каналов 1 и количеством выходных каналов 32. Данный слой проходит через функцию активации ReLU.
* Слой пулинга MaxPool1d с размером ядра 2.
* Слой свертки Conv1d с ядром размера 3, количеством входных каналов 32 и количеством выходных каналов 64. Данный слой проходит через функцию активации ReLU.
* Еще один слой пулинга MaxPool1d с размером ядра 2.
* Слой преобразования размерности (view), который изменяет форму тензора на [batch\_size, 64\*4].
* Полносвязный слой Linear с количеством входных признаков 64\*4 и количеством выходных признаков 128. Данный слой проходит через функцию активации ReLU.
* Еще один полносвязный слой Linear с количеством входных признаков 128 и количеством выходных признаков 1. Данный слой является выходным слоем модели.



Рисунок 20 – Третья модель

Также определены функция потерь и оптимизатор. В третьей модели функция потерь – MSELoss, а оптимизатор – Adam (рисунок 21).

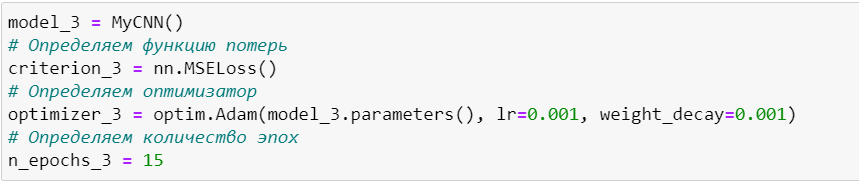


Рисунок 21 - Функция потерь, оптимизатор третьей модели

Обучение происходит 3 модели происходит аналогично 1 модели (рисунок 22).



Рисунок 22 – Обучение 3 модели

После обучения модели нужно провести анализ результатов обучения (рисунок 23). Для оценки качества третьей модели я использовал такие метрики качества как: R2, MSE, MAE. В целом, значения метрик R2, MSE и MAE указывают на высокое качество работы модели. Ошибки третьей модели ниже, чем у первой и второй.

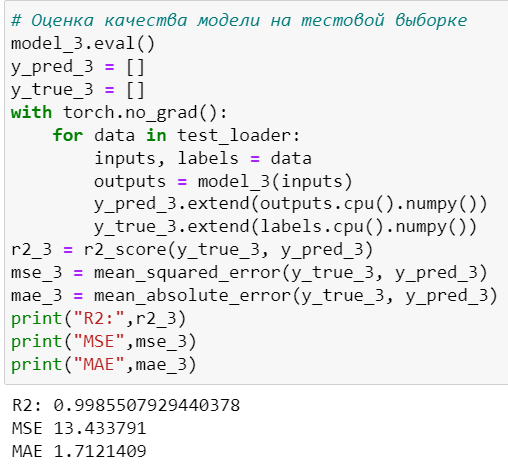


Рисунок 23 - Оценка качества третьей модели

На рисунке 24 представлено обучение 3 модели на обучающей и валидационной выборке.

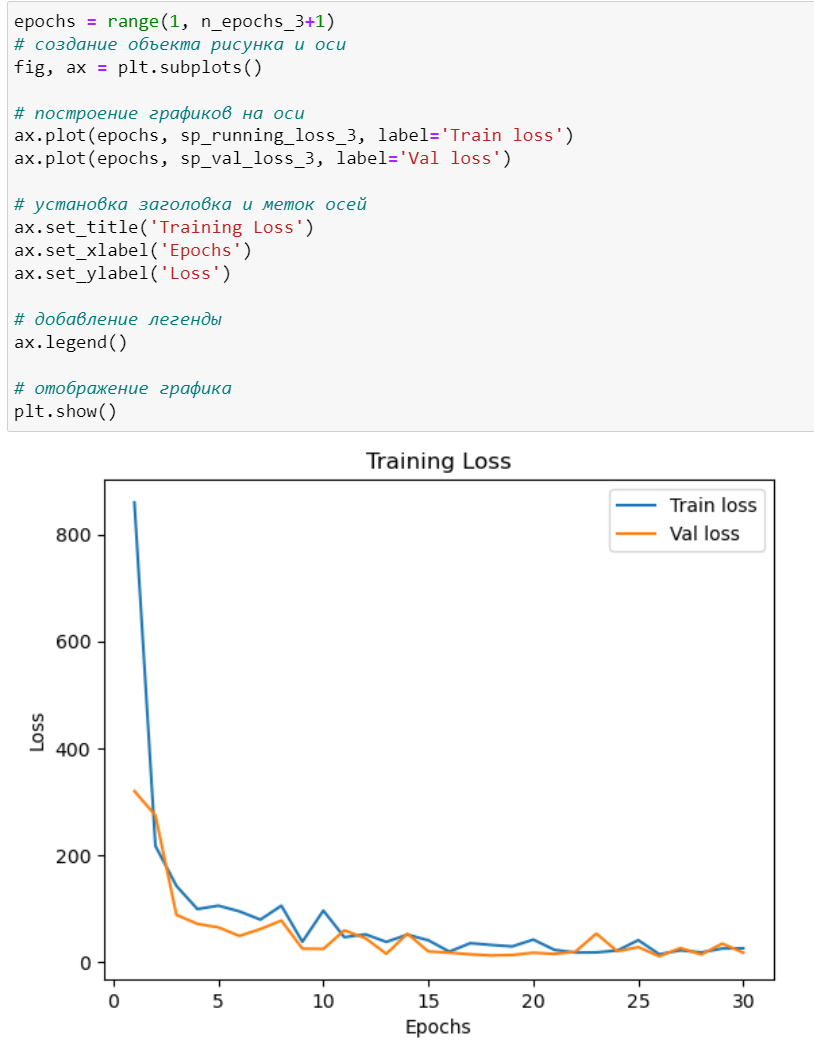


Рисунок 24 – График обучения 3 модели

### 2.3.4 Построение, обучение и анализ 4 модели

Данная модель представляет собой простую DNN (Deep Neural Network) с 6 полносвязными слоями (Fully Connected Layers) для задачи регрессии (рисунок 25).

Архитектура модели:

* Входной слой: принимает тензор размерности (batch\_size, 16), где 16 - количество признаков
* Скрытый слой 1: полносвязный слой с 32 нейронами
* Функция активации 1: ReLU
* Скрытый слой 2: полносвязный слой с 64 нейронами
* Функция активации 2: ReLU
* Скрытый слой 3: полносвязный слой с 32 нейронами
* Функция активации 3: ReLU
* Скрытый слой 4: полносвязный слой с 16 нейронами
* Функция активации 4: ReLU
* Скрытый слой 5: полносвязный слой с 8 нейронами
* Функция активации 5: ReLU
* Выходной слой: полносвязный слой с 1 нейроном (выходом модели)

Все полносвязные слои (fc1 - fc6) имеют bias-вектора. Функция активации на каждом скрытом слое (sig1 - sig5) - это ReLU (Rectified Linear Unit), а на выходном слое функция активации не используется, поскольку мы ожидаем непрерывный выход.

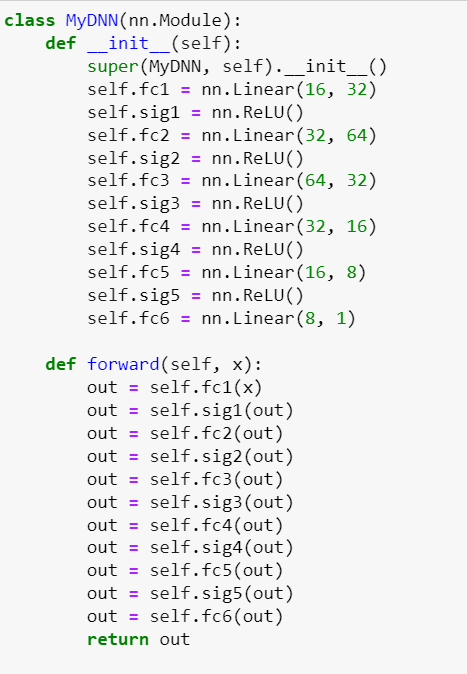


Рисунок 26 – Четвертая модель

На рисунке 27 определены функция потерь и оптимизатор. В третьей модели функция потерь – MSELoss, а оптимизатор – Adam.

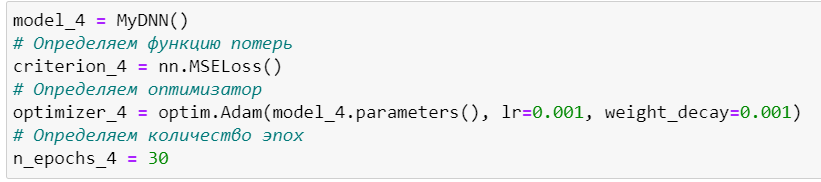


Рисунок 27 – Определение функций потерь, оптимизатора и количества эпох

Обучение происходит 3 модели происходит аналогично 1 модели (рисунок 28).



Рисунок 28 – Обучение 4 модели

На рисунке 29 показаны оценки качества четвертой модели.

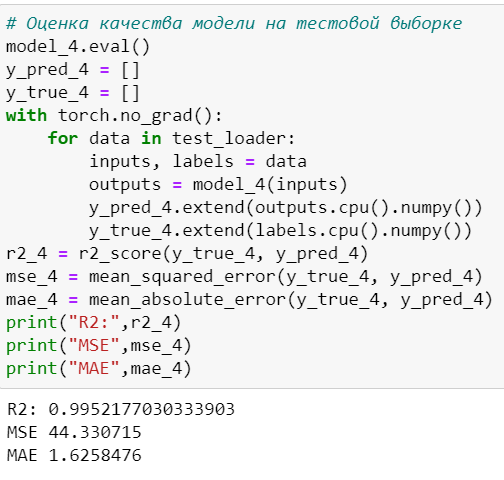


Рисунок 29 – Оценки качества 4 модели

Чтобы посмотреть, как обучалась модель по эпохам на обучающей и валидационной выборке я построил график, который отображает изменение значения функции потерь (loss) в процессе обучения модели на обеих моделях (рисунок 30).



Рисунок 30 - График изменения значения функции потерь четвертой модели

## **2.4 Сравнение моделей**

Выведем все метрики оценки качества каждой модели в виде графиков (рисунок 31) и таблицы (рисунок 32)

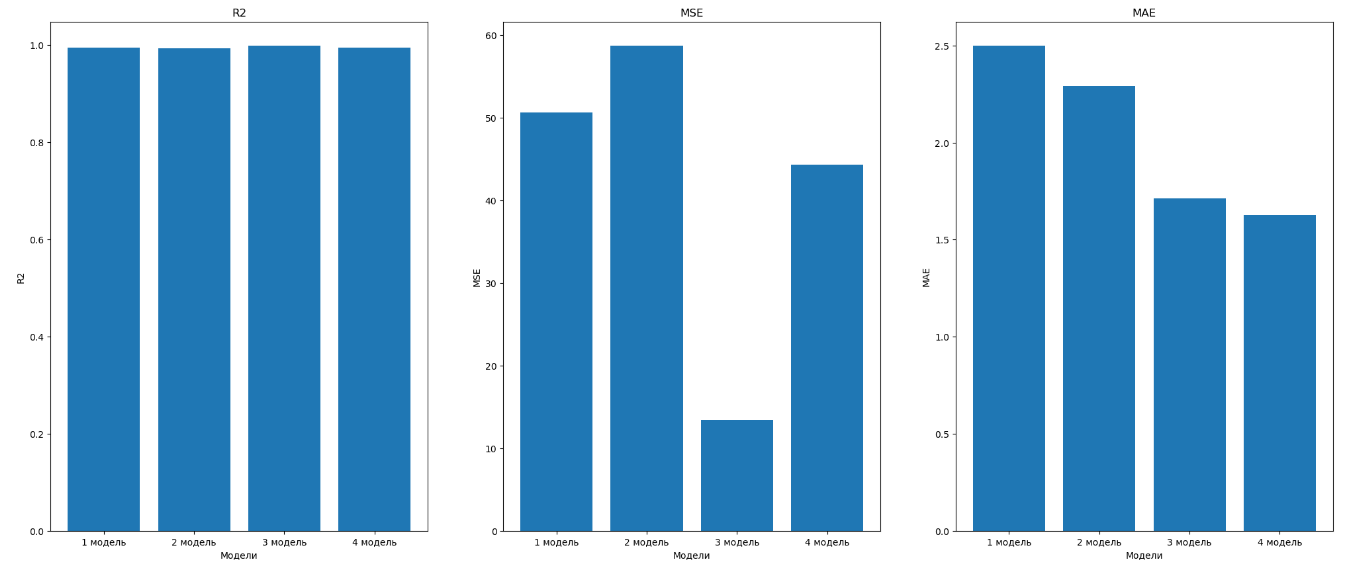


Рисунок 31 - Метрики оценки качества каждой модели

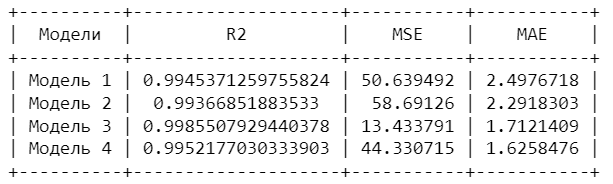


Рисунок 32 – Таблица метрик качества моделей

Из этих данных можно выделить, что все модели смогли адаптироваться к данным и делать более-менее релевантные прогнозы. Но лучше всего обучилась 3 модель (CNN).

# **Глава 3. Заключение**

В заключение, можно сказать, что применение глубокого обучения для прогнозирования и анализа выручки предприятия является весьма перспективным подходом. В процессе работы были изучены основы глубокого обучения, различные типы архитектур и методы их обучения.

Поставленная цель разработать модели нейронных сетей для прогнозирования выручки предприятия была достигнута. Все задачи были выполнены. В ходе работы я изучил методы прогнозирования и анализа выручки предприятия с помощью нейронных сетей, проанализировал данные о выручке предприятия, разработал и обучи несколько моделей нейронных сетей, оценил их точность.

Полученные результаты могут быть использованы для оптимизации процессов управления предприятием и принятия решений. Однако, для дальнейшего улучшения прогнозных моделей необходимо провести дополнительные исследования и эксперименты, в том числе с использованием более разнообразных данных и более сложных моделей.

Таким образом, использование глубокого обучение для прогнозирования выручки предприятия может быть весьма полезным инструментом для улучшения эффективности бизнес-процессов и принятия взвешенных решений.

# **Список использованных источников**

Основные источники:

1. <https://pytorch.org/docs/stable/index.html>
2. <https://scikit-learn.org/stable/index.html>
3. <https://pandas.pydata.org>
4. Макмахан Брайан, Рао Делип. Знакомство с PyTorch: глубокое обучение при обработке естественного языка. — СПб.: Питер, 2020. — 256 с.:
5. Эли Стивенс, Томас Вихманн. Глубокое обучение с PyTorch
6. <https://www.kaggle.com/datasets/sai1881/revenue>
7. Себастьян Рашка. Python Machine Learning – 2015

Дополнительные источники:

1. L.P. Coelho, W. Richert. Building machine learning systems with Python - 2015 (326p)
2. Joe Papa. PyTorch Pocket Reference – 2021