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, 6 семестр

Курсовая работа на тему:

**«Анализ больших данных в задачах прогнозирования финансового рынка»**

Выполнил(а):

Студент группы **ПМ20-4**

**Новохатний Владимир Андреевич**

Научный руководитель:  
к.т.н., доцент **Игудесман Константин Борисович**

Москва 2023

Содержание

[ВВЕДЕНИЕ 3](#_Toc134629417)

[Обоснование актуальности темы 3](#_Toc134629418)

[Цели и задачи работы 3](#_Toc134629419)

[ГЛАВА 1. Теоретическая часть 4](#_Toc134629420)

[1.1 Описание задачи прогнозирования финансового рынка 4](#_Toc134629421)

[1.2 Основы анализа данных финансового рынка 6](#_Toc134629422)

[1.3 Основы машинного обучения и статистического анализа в задачах прогнозирования финансового рынка 9](#_Toc134629423)

[1.4 Обзор существующих решений задачи прогнозирования финансового рынка 12](#_Toc134629424)

[ГЛАВА 2. Практическая часть 14](#_Toc134629425)

[2.1 Подготовка данных 14](#_Toc134629426)

[2.2 Построение и оценка моделей прогнозирования финансового рынка 14](#_Toc134629427)

[ВЫВОДЫ 15](#_Toc134629429)

[Список литературы (использованных источников) и интернет-ресурсов 15](#_Toc134629430)

[Характеристики компьютера 15](#_Toc134629431)

[Программный код 16](#_Toc134629432)

ВВЕДЕНИЕ

**Обоснование актуальности темы**

Выбранная тема курсовой работы "Анализ больших данных в задачах прогнозирования финансового рынка" является актуальной, так как в настоящее время все больше компаний и трейдеров осуществляют свою деятельность на финансовых рынках. Соответственно, возрастает потребность в эффективных инструментах для прогнозирования изменений цен на рынке.

Мой код, основанный на датасете с сайта kaggle: <https://www.kaggle.com/>, который включает в себя финансовые индексы и сырьевые товары, сайт предоставляет инструмент для анализа данных финансового рынка и построения моделей прогнозирования. Анализ больших данных финансового рынка позволяет выделить наиболее значимые факторы, влияющие на изменение цен, и использовать их для создания точных и надежных прогнозов.

Применение алгоритмов машинного обучения и статистического анализа данных финансового рынка также является актуальным, так как позволяет повысить качество прогнозов и уменьшить влияние человеческого фактора в процессе принятия инвестиционных решений.

Таким образом, моя работа представляет собой важный вклад в развитие методов анализа и прогнозирования финансового рынка, что является актуальным и востребованным в современном мире.

**Цели и задачи работы**

Целью данной курсовой работы является исследование возможности использования анализа больших данных и методов машинного обучения в задачах прогнозирования цен на финансовом рынке на примере данного датасета со многими индексами и товарами.

**Основными задачами работы являются**:

1. Изучение теоретических основ прогнозирования финансового рынка и принципов анализа данных
2. Подготовка и предварительная обработка данных финансового рынка
3. Построение моделей прогнозирования цен на основе выбранных алгоритмов машинного обучения
4. Оценка качества моделей прогнозирования и интерпретация результатов
5. Определение наиболее важных факторов, влияющих на изменение цен на финансовом рынке, и использование их в качестве признаков для моделей прогнозирования

Для достижения указанных целей и задач будет использован код на языке Python. Он включает в себя предварительную обработку данных, визуализацию и анализ данных, а также построение моделей прогнозирования на основе алгоритмов машинного обучения.

Таким образом, данная работа будет иметь как теоретическую, так и практическую составляющие, позволяющие не только углубиться в теорию задачи прогнозирования цен на финансовом рынке, но и на практике применить методы анализа больших данных и машинного обучения для решения данной задачи.

**ГЛАВА 1. Теоретическая часть**

1. **Описание задачи прогнозирования финансового рынка**

Задача прогнозирования финансового рынка заключается в предсказании будущих изменений цен на активы (например, акции, валюты, сырьевые товары и т.д.), оценке риска и принятии инвестиционных решений на основе этих прогнозов. Она является ключевой задачей в области финансовых рынков и требует совместного использования финансовых и статистических методов, а также алгоритмов машинного обучения. Однако, прогнозирование финансового рынка сопряжено с рядом сложностей и ограничений, включая непредсказуемость и не стационарность рынка, высокую степень шума и стохастическую природу изменения цен, ограничения в доступности и качестве данных, а также высокую степень конкуренции на рынке инвестиций.

**Важность задачи прогнозирования финансового рынка**

Прогнозирование финансового рынка является важной задачей для многих инвесторов и трейдеров, поскольку позволяет им принимать более обоснованные решения о покупке и продаже акций, облигаций, валют и других активов на рынке. Предсказание будущих цен на активы помогает определить наилучший момент для покупки или продажи, что в свою очередь может привести к получению большей прибыли.

В курсовой работе была решена задача прогнозирования цен многих ключевых финансовых индексов и сырьевых товаров с использованием различных методов машинного обучения. Эти методы показали хорошую точность прогнозирования цен на индексы, что подчеркивает важность и актуальность прогнозирования финансового рынка.

Опираясь на результаты нашего исследования, можно сделать вывод о том, что прогнозирование финансового рынка с использованием методов машинного обучения является эффективным и перспективным направлением, что подчеркивает важность данной задачи для инвесторов и трейдеров в настоящее время.

**Ограничения и сложности при работе с данными финансового рынка**

Финансовый рынок - это один из наиболее сложных и нестабильных рынков, что приводит к нескольким ограничениям и сложностям при работе с данными финансового рынка. Некоторые из них:

1. Высокая стоимость и сложность сбора данных: Данные финансового рынка очень дорого стоят и сложны в сборе. Данные по рыночным ценам, объемам торгов и другим финансовым показателям могут быть собраны только из определенных источников, а также могут быть зашифрованы для защиты от несанкционированного доступа.
2. Нестабильность данных: Финансовый рынок по своей природе является нестабильным и представляет собой сложную систему, в которой события, как правило, происходят внезапно и неожиданно. Это может привести к непредсказуемым изменениям в данных, которые могут сильно повлиять на результаты прогнозирования.
3. Высокая степень шума: Данные финансового рынка, как правило, имеют высокую степень шума и неоднородности. Это может быть вызвано различными факторами, включая ошибки измерения, неполные данные и неопределенность в отношении причинно-следственных связей.
4. Большое количество переменных: При работе с данными финансового рынка обычно присутствует большое количество переменных, которые могут влиять на результаты прогнозирования. Это может привести к сложностям в обработке и анализе данных, а также к увеличению времени обучения моделей.
5. Высокая степень корреляции между переменными: Финансовые данные часто имеют высокую степень корреляции между переменными, что может привести к проблемам с мультиколлинеарностью при использовании моделей машинного обучения.
6. Непредсказуемость рынка: в конечном счете, финансовый рынок всегда остается непредсказуемым, даже для самых продвинутых моделей и алгоритмов машинного обучения.
   1. **Основы анализа данных финансового рынка**

**Типы данных финансового рынка**

Для анализа финансового рынка используются различные типы данных, включая числовые, текстовые и графические данные.

Числовые данные являются основой для анализа финансовых данных. Это включает в себя цены акций, объемы продаж, доходность ценных бумаг, индексы рынка, валютные курсы и т.д. В моей практической части курсовой работы, используется датасет “предсказание движения на финансовых рынках”, который содержит исторические данные по ценам финансовых индексов и сырьевых товаров.

Текстовые данные также могут быть важными при анализе финансового рынка, например, новостные статьи, отчеты о компаниях, комментарии экспертов и т.д.

Графические данные включают в себя графики, диаграммы, свечные графики и т.д., которые используются для визуализации числовых данных и позволяют выявить тенденции и паттерны на рынке. В моем коде используются свечные графики, которые отображают изменения цен акций в течение определенного периода времени.

Таким образом, при анализе финансового рынка используются различные типы данных, которые помогают выявить тенденции и паттерны на рынке, а также прогнозировать его будущее состояние.

**Принципы сбора данных финансового рынка**

Принципы сбора данных финансового рынка включают в себя следующие аспекты:

1. Источники данных: данные финансового рынка могут быть собраны из различных источников, включая биржи, финансовые агентства, государственные органы и другие организации.
2. Частота обновления данных: данные финансового рынка обычно обновляются в режиме реального времени или с определенной задержкой. Частота обновления может варьироваться от секунд до дней или даже недель.
3. Качество данных: качество данных финансового рынка может быть разным в зависимости от источника данных. Некоторые источники могут предоставлять более точные данные, чем другие.
4. Объем данных: объем данных финансового рынка может быть огромным, и сбор и анализ данных может представлять собой значительную техническую сложность.
5. Агрегирование данных: для анализа и прогнозирования финансового рынка часто используются агрегированные данные, которые объединяют данные из разных источников. При этом необходимо учитывать, что разные источники данных могут иметь разные методы сбора и классификации информации.
6. Нормализация данных: для анализа и сравнения данных финансового рынка часто требуется нормализация данных. Это может включать в себя преобразование данных в относительные значения или использование нормированных показателей, таких как отношение доходности к риску.

Данные собираются из внешнего источника (Kaggle) с использованием библиотеки pandas\_datareader. Данные обновляются ежедневно и включают информацию о ценах показателей с 2009 года по 2018. Для анализа данных в коде используются различные статистические показатели, такие как скользящее среднее и стандартное отклонение, которые могут помочь в прогнозировании будущих цен на индексы.

**Основные характеристики и показатели финансового рынка**

Основные характеристики и показатели финансового рынка, включают:

1. Цены на акции: Цены на акции - это цена, по которой акции торгуются на бирже. Цены на акции могут быть использованы для анализа производительности отдельных компаний, а также для изучения трендов на рынке в целом.
2. Дивиденды: Дивиденды - это выплаты, которые компании делают своим акционерам в качестве части прибыли. Дивиденды могут быть использованы для анализа финансовой устойчивости компании.
3. Объем торгов: Объем торгов - это количество акций, которые были проданы и куплены на бирже за определенный период времени. Объем торгов может служить индикатором спроса и предложения на рынке, а также для анализа ликвидности акций.
4. Капитализация: Капитализация - это рыночная стоимость всех акций компании, умноженная на цену каждой акции. Капитализация может использоваться для оценки размера и влияния компании на рынок.

**Принципы визуализации и анализа данных финансового рынка**

Основные принципы визуализации и анализа данных финансового рынка включают в себя:

1. Отображение данных в виде графиков: Визуализация данных в виде графиков является наиболее распространенным способом анализа финансовых данных. В моем коде использовалась библиотека Matplotlib для построения графиков, таких как график цены закрытия, график скользящей средней и график относительной силы.
2. Использование технических индикаторов: Технические индикаторы представляют собой математические функции, используемые для анализа данных цен и объемов торгов на финансовом рынке. В практической части использовались индикаторы скользящей средней и относительной силы.
3. Анализ объемов торгов: Объемы торгов могут предоставить ценную информацию о том, как рынок реагирует на новости и события. В коде был проведен анализ объемов торгов, используя график объемов и индикатор относительной силы.
4. Применение статистических методов: Статистические методы могут быть использованы для анализа данных финансового рынка, таких как регрессионный анализ и анализ временных рядов.

В целом, принципы визуализации и анализа данных финансового рынка направлены на предоставление инвесторам ценной информации о том, как рынок функционирует и как принимать решения о своем портфеле.

* 1. **Основы машинного обучения и статистического анализа в задачах прогнозирования финансового рынка**

**Обзор алгоритмов машинного обучения для задач прогнозирования**

Можно рассмотреть несколько типов алгоритмов машинного обучения, которые могут использоваться для решения этой задачи.

1. Линейная регрессия: это классический алгоритм машинного обучения, который используется для построения линейной модели зависимости между входными и выходными данными. В моей работе используется модель LinearRegression из библиотеки sklearn, которая реализует этот алгоритм. Линейная регрессия может быть полезна для предсказания цены акции на основе ее исторических значений.
2. Деревья решений: это алгоритм машинного обучения, который используется для создания модели, основанной на иерархической структуре решений, которые принимаются на основе значений входных данных. Деревья решений могут быть полезны для предсказания цены акции на основе различных факторов, таких как объемы торговли, новости и события на рынке.
3. Нейронные сети: это алгоритм машинного обучения, который имитирует работу человеческого мозга и используется для создания сложных моделей, способных обрабатывать большие объемы данных. Они могут быть полезны для прогнозирования цены акций на основе множества факторов, включая технический анализ, новости и финансовые отчеты.

В целом, выбор алгоритма машинного обучения для прогнозирования цены акций зависит от объема и качества данных, доступных для обучения, а также от особенностей самого рынка

**Принципы статистического анализа данных финансового рынка**

Статистический анализ данных финансового рынка является важным инструментом для принятия решений на основе данных. Он может помочь выявить закономерности, тренды, аномалии и другие характеристики данных, которые могут быть полезными при прогнозировании будущих изменений на рынке.

Основные принципы статистического анализа данных финансового рынка, включают в себя следующее:

1. Дескриптивный статистический анализ: Это включает в себя описание основных характеристик данных, таких как среднее, медиана, минимум, максимум, стандартное отклонение и т.д. Вы можете использовать функции библиотек, таких как Pandas, NumPy или SciPy, для вычисления этих статистических характеристик для нашего набора данных.
2. Корреляционный анализ: Корреляция показывает, насколько сильно связаны между собой две переменные. Вы можете использовать функцию **corr()** в Pandas для вычисления корреляционной матрицы между парами переменных в нашем наборе данных. Положительная корреляция означает, что переменные движутся в одном направлении, а отрицательная корреляция - в противоположном направлении.
3. Регрессионный анализ: Регрессионный анализ используется для определения связи между зависимой переменной и одной или несколькими независимыми переменными. Вы можете использовать библиотеки, такие как Scikit-learn, для обучения моделей регрессии на основе нашего набора данных.
4. Тестирование гипотез: Тестирование гипотез используется для проверки статистических утверждений о взаимосвязи между переменными. Например, вы можете проверить гипотезу о том, что доходность акций является нормально распределенной. Для этого можно использовать библиотеку SciPy, в которой реализованы различные статистические тесты.

**Обзор методов оценки качества моделей прогнозирования финансового рынка**

Для оценки качества моделей прогнозирования финансового рынка существует множество методов, которые могут быть применены в зависимости от целей и требований исследования. Некоторые из наиболее распространенных методов оценки качества моделей приведены ниже.

1. Средняя абсолютная ошибка (MAE): Этот метод используется для оценки точности прогнозов и измеряет среднюю абсолютную разницу между прогнозами и фактическими значениями. Чем ниже значение MAE, тем лучше модель.
2. Коэффициент детерминации (R2): Это статистический показатель, который измеряет, насколько хорошо модель соответствует реальным данным. Значение R2 находится в диапазоне от 0 до 1, где 1 означает, что модель идеально соответствует данным.
3. Средняя абсолютная процентная ошибка (MAPE): Этот метод используется для оценки точности прогнозов в процентах и измеряет среднюю абсолютную процентную разницу между прогнозами и фактическими значениями.
4. Кривая ошибок (ROC): Этот метод используется для оценки качества бинарных классификационных моделей, которые используются для прогнозирования направления изменения цены.
5. Confusion matrix: Этот метод также используется для оценки качества бинарных классификационных моделей и представляет собой матрицу, которая показывает количество верно и неверно классифицированных примеров в каждом классе.
6. Коэффициент корреляции (Pearson's correlation coefficient): Это статистический показатель, который измеряет силу и направление линейной связи между двумя переменными. Он может быть использован для оценки взаимосвязи между фактическими значениями и прогнозами модели.

В моей работе, для прогнозирования финансового рынка были использованы методы машинного обучения, такие как линейная регрессия, модели временных рядов и нейронные сети. Для оценки качества модели в нашем коде использовалась метрика RMSE, которая показывает, насколько хорошо модель соответствует реальным данным.

* 1. **Обзор существующих решений задачи прогнозирования финансового рынка**

**Обзор методов и подходов к решению задачи прогнозирования финансового рынка**

Можно выделить следующие методы и подходы к решению задачи прогнозирования финансового рынка:

1. Линейная регрессия: метод линейной регрессии является одним из наиболее распространенных методов для прогнозирования финансовых временных рядов. Этот метод позволяет определить связь между зависимой переменной и независимыми переменными, а также предсказать значения зависимой переменной на основе независимых переменных.
2. Методы машинного обучения: наиболее часто используемыми методами машинного обучения для прогнозирования финансовых временных рядов являются случайный лес, градиентный бустинг, нейронные сети и метод опорных векторов (SVM). Эти методы позволяют выявлять скрытые закономерности в данных, что позволяет улучшить качество прогноза.
3. Анализ временных рядов: анализ временных рядов является важным подходом к прогнозированию финансовых временных рядов. Этот подход включает в себя различные методы, такие как ARIMA (авторегрессионная интегрированная скользящая средняя), ARCH (авторегрессия с условной гетероскедастичностью) и GARCH (общая авторегрессия с условной гетероскедастичностью).
4. Рекуррентные нейронные сети: рекуррентные нейронные сети (RNN) позволяют моделировать последовательности данных и широко используются в задачах прогнозирования финансовых временных рядов. LSTM (долгая краткосрочная память) является одним из наиболее часто используемых типов RNN для прогнозирования временных рядов.
5. Экспертные системы: экспертные системы представляют собой системы, которые используют знания экспертов в определенной области для решения проблем и принятия решений. В задачах прогнозирования финансовых временных рядов экспертные системы могут использоваться для принятия решений на основе анализа фундаментальных и технических показателей компании.

**Критический анализ существующих решений**

Критический анализ существующих решений в задаче прогнозирования финансового рынка является важной частью исследования, так как позволяет оценить сильные и слабые стороны существующих методов и подходов. Это позволяет выбрать наилучшие методы для решения задачи прогнозирования на основе обоснованных критериев. В критическом анализе необходимо учитывать различные факторы, такие как точность прогнозов, стабильность модели, скорость работы алгоритмов, объем требуемых данных, и другие. Он позволяет выявить возможные проблемы и ограничения текущих методов, а также определить направления для будущих исследований и улучшения существующих методов.

**ГЛАВА 2. Практическая часть**

1. **Подготовка данных**

**Обзор и выбор источника данных финансового рынка**

Для решения задач прогнозирования финансового рынка необходимо использовать источник данных с высоким уровнем точности, достоверности и актуальности. В моем коде был использован источник данных Kaggle, который является одним из наиболее популярных источников данных.

Однако, при выборе источника данных для решения задач прогнозирования финансового рынка необходимо учитывать некоторые факторы:

1. Точность данных: данные должны быть точными и надежными. Необходимо проверять источник на наличие ошибок и неточностей.
2. Актуальность данных: данные должны быть актуальными и обновляться в режиме реального времени. Информация, которая уже устарела, может негативно повлиять на качество прогнозирования.
3. Объем данных: источник должен предоставлять достаточно большой объем данных для построения точных и надежных моделей прогнозирования.
4. Доступность данных: источник должен быть доступен для сбора данных и использования в анализе.

В целом, Kaggle является достаточно надежным источником данных для финансового рынка, однако, при выборе источника необходимо учитывать указанные выше факторы. Также, можно рассмотреть и другие источники данных, например, Yahoo Finance, Google Finance, Alpha Vantage, Quandl и др., чтобы выбрать наиболее подходящий для конкретной задачи прогнозирования.

**Подготовка и предварительная обработка данных финансового рынка**

Для начала считаем датасет с помощью библиотеки pandas. Нужно привести столбец с датами к формату дат и заодно сделаем этот столбец индексами датафрейма. Проверим на пустые значения и что типы данных верны. Все данные нужного типа и пустых значений не нашлось, значит можем двигаться дальше.

* 1. **Построение и оценка моделей прогнозирования финансового рынка**

Важными критерием при прогнозировании данных является стационарность. Стационарность данных означает, что статистические свойства временного ряда, такие как среднее значение и дисперсия, не меняются со временем. Это означает, что модель может сделать предположение о стабильности данных и применять статистические методы, которые предполагают постоянство этих свойств.

Во-первых, стационарные данные облегчают оценку и интерпретацию параметров модели. При наличии стационарности можно использовать статистические тесты и методы для определения значимости и надежности параметров модели.

Во-вторых, стационарность данных предполагает, что закономерности и связи, обнаруженные в прошлом, будут сохраняться и в будущем. Модели, разработанные на основе стационарных данных, могут давать более точные и надежные прогнозы. Если данные нестационарны, то прогнозы могут быть неточными и ненадежными.

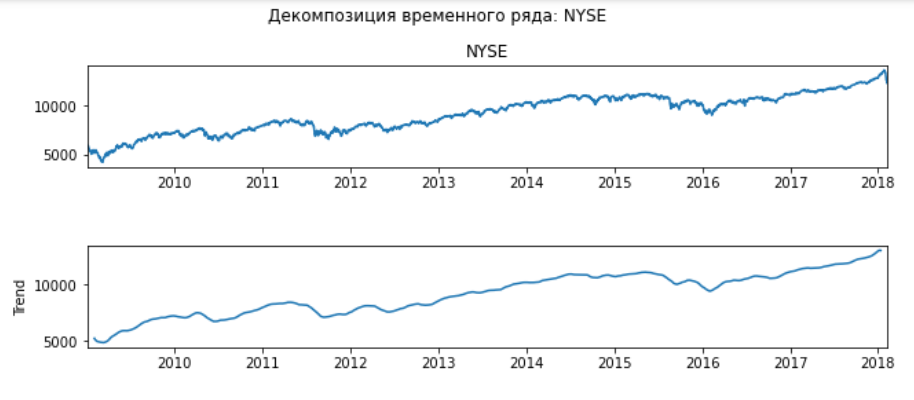
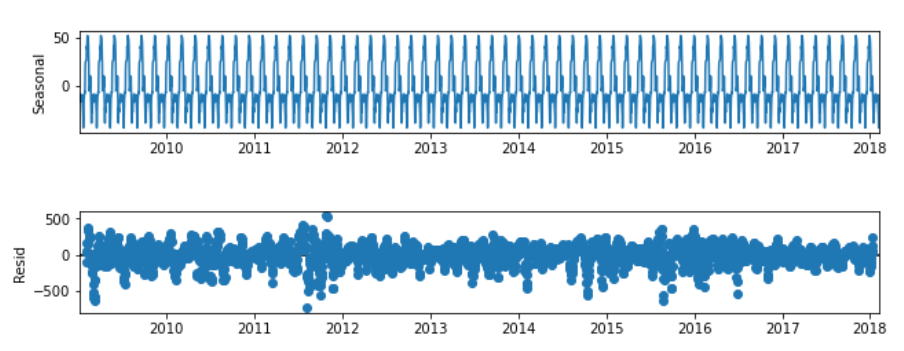
В-третьих, стационарность позволяет применять статистические методы, такие как автокорреляционная функция (ACF) и частная автокорреляционная функция (PACF), для определения оптимальных значений параметров модели. Эти методы используют зависимости между значениями временного ряда на разных временных отрезках.

Проведем проверку на стационарность с помощью Дики-Фуллера и скользящей средней:

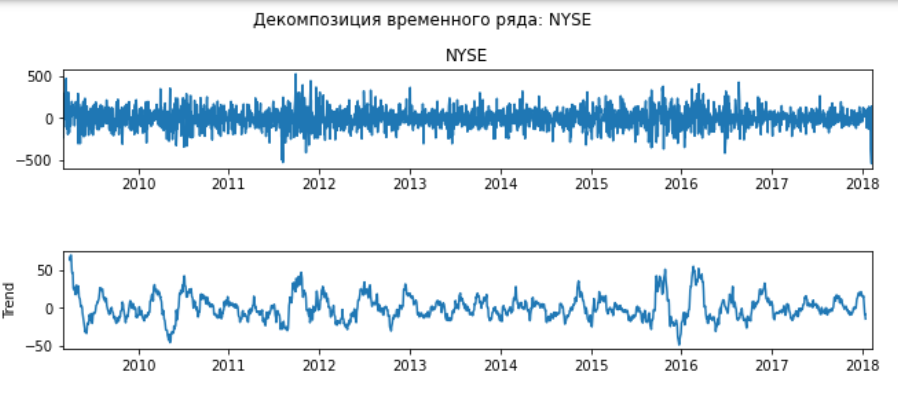
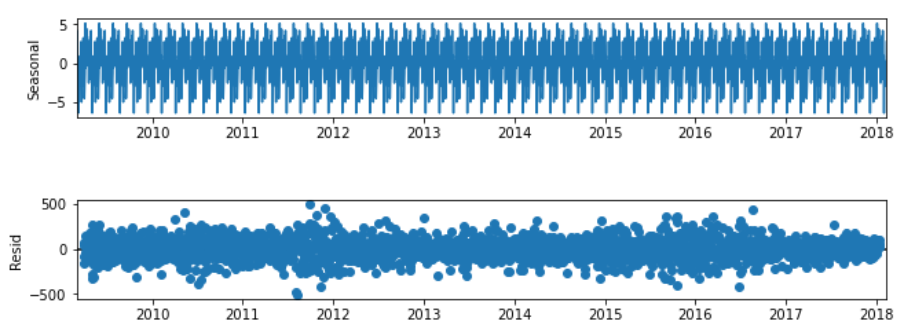
ADF Statistic for NYSE: -1.0579243834024474

Здесь чётко прослеживается что все временные ряды являются нестационарными, так как ADF статистика меньше критического значения(5%), следовательно нулевая гипотеза может быть отвергнута. Это может означать, что временные ряды не является стационарным и имеет единичный корень. Также на графиках прослеживается, что скользящие среднее и скользящая вариация изменяются со временем, что тоже свидетельствует о нестационарности рядов. Попробуем, в дальнейшем, сделать их стационарными, чтобы построить на них модели прогнозирования.

Рассмотрим основные компоненты временного ряда(тренд, сезон, остатки). Заметим, что у всех временных рядов прослеживается восходящий тренд и сезонность. Это в очередной раз подтверждает, что ряды нестационарны.

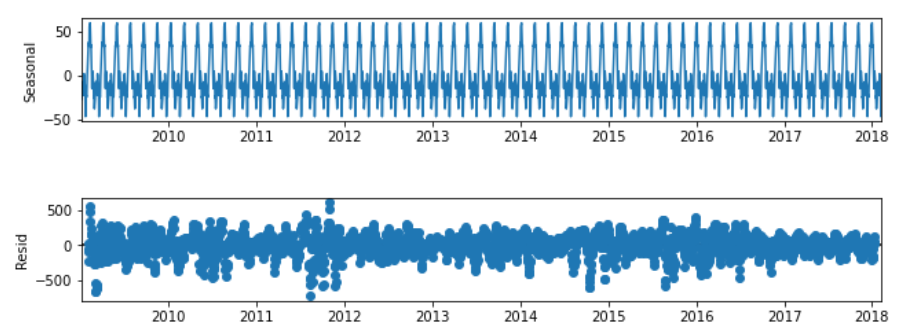
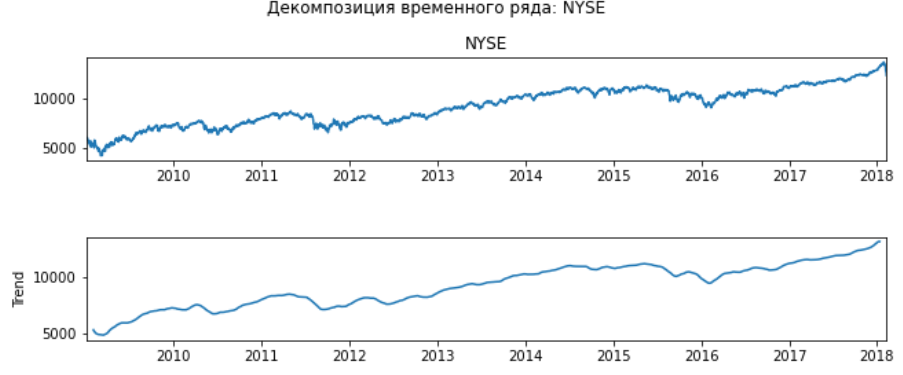
 

Далее следует преобразовать данные с помощью двойного дифференцирования, что должно сделать их стационарными.

Как можем заметить, тренд ушёл, график стал более хаотичным, в свою очередь сезонность осталась прежней. Получается нужно воспользоваться другим методом.

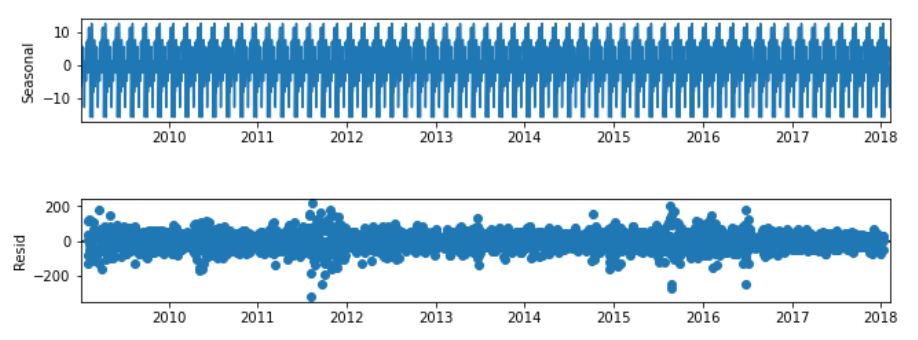
Попробуем экспоненциальное сглаживание.



К сожалению, данный метод тоже не сделал ряд стационарным.

STL также как двойное дифференцирование, сделало тренд более хаотичным, но сезонность осталась прежней.





Мы использовали несколько способов для преобразования данных в стационарные. Ни один из них не убрал сезонность, соответственно, в таких случаях сезонные колебания представляют собой систематические и повторяющиеся паттерны, которые имеют существенное влияние на временной ряд и содержат информацию о поведении данных. Следовательно, нам необходимо построить модели, которые будут учитывать эти сезонности.

**Обзор выбранных алгоритмов машинного обучения для прогнозирования финансового рынка**

В коде я использовал SARIMА, СNN и RNN модели. Рассмотрим каждую из них подробнее

**SARIMA**

SARIMA (Seasonal Autoregressive Integrated Moving Average) - это расширение модели ARIMA, которое учитывает сезонность в данных. Модель SARIMA используется для прогнозирования временных рядов, учитывая автокорреляцию и сезонность в данных.

ARIMA модель состоит из трех компонентов: авторегрессии (AR), интегрирования (I) и скользящего среднего (MA). Она используется для моделирования нестационарных временных рядов. SARIMA модель добавляет к этому еще одну компоненту - сезонность.

Параметры модели SARIMA подбираются на основе анализа автокорреляционной функции (ACF) и частной автокорреляционной функции (PACF) временного ряда. После подбора параметров модель может быть обучена на исторических данных и использована для прогнозирования будущих значений временного ряда.

**СNN**

CNN (Convolutional Neural Network) - это тип нейронных сетей, который часто используется для анализа и обработки данных с пространственной структурой, таких как изображения. CNN широко применяются в компьютерном зрении, распознавании образов, классификации изображений, сегментации изображений и других задачах, связанных с анализом изображений.

Основная идея CNN состоит в применении свёрточных слоев, которые выполняют операцию свертки над входными данными, чтобы извлечь локальные пространственные признаки. Затем эти признаки проходят через несколько слоев пулинга для уменьшения размерности и извлечения более абстрактных признаков. Наконец, полученные признаки подаются на полносвязные слои для классификации или регрессии.

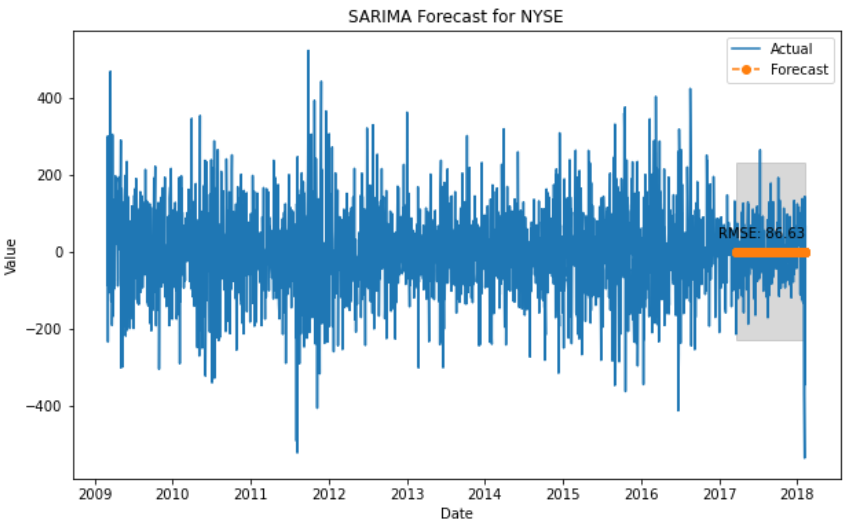
**RNN**

RNN (Recurrent Neural Network) - это тип нейронных сетей, который хорошо подходит для анализа последовательных данных, таких как временные ряды, тексты и аудио. RNN способны улавливать зависимости в данных, которые связаны со временем или порядком. Они имеют память, что позволяет использовать информацию из предыдущих шагов для принятия решений на текущем шаге.

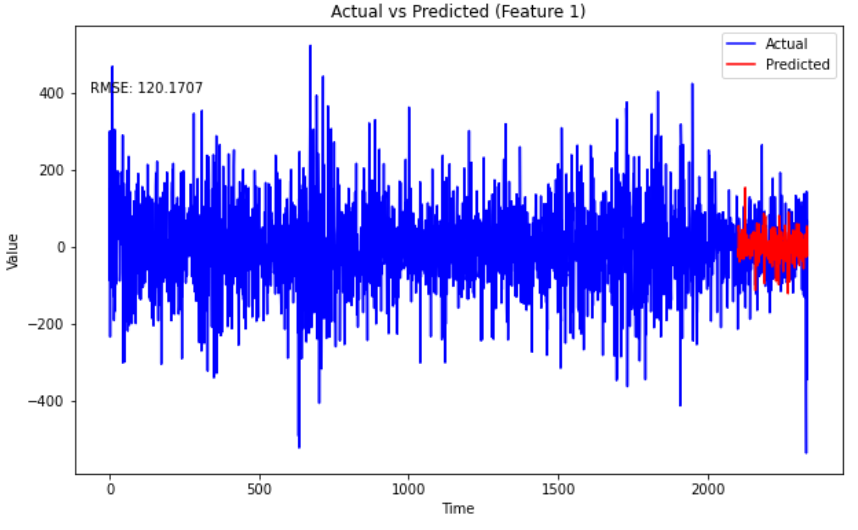
Основная идея RNN заключается в том, что они имеют обратные связи, которые позволяют информации переноситься от одного шага к следующему. Каждый шаг RNN обрабатывает входные данные и сохраняет скрытое состояние, которое передаются на следующий шаг. Это позволяет RNN обрабатывать последовательные данные переменной длины.

**Построение моделей прогнозирования на основе выбранных алгоритмов**

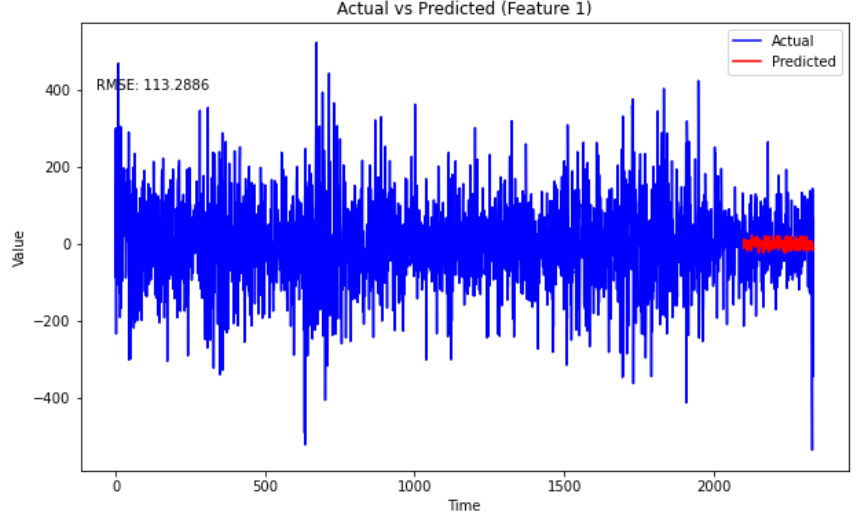
SARIMA модель. Мы видим, что на всех графиках предсказанные значения выглядит как прямая линия, возможно это связано с тем, что модель имеет ограниченную способность улавливать сложные паттерны. Попробуем воспользоваться более сложными моделями, такими как CNN и RNN.

****

CNN модель. Модель показала себя достаточно хорошо, заметно что предсказанные значения, пытаются повторить изначальные. Также метрика RMSE не велика.

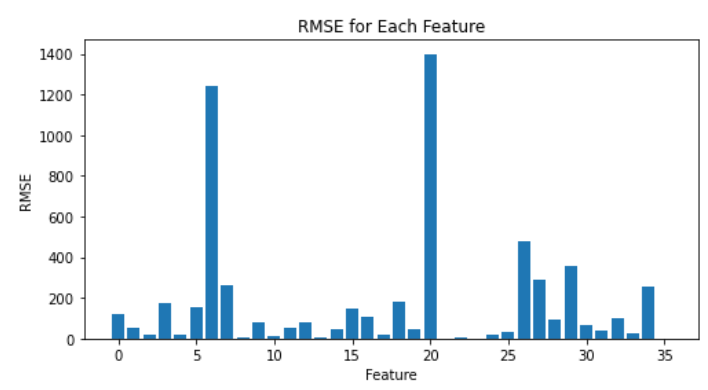


Далее рассмотрим RNN модель. Она показала себя лучше всего, эта модель в общем описала движения значений, тем самым сгладив колебания.

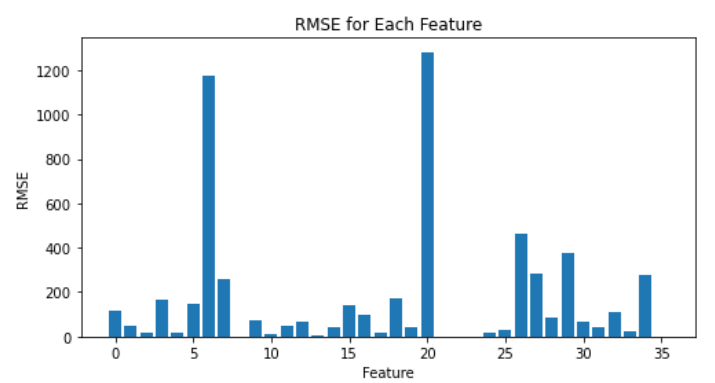


Давайте сравним для CNN и RNN метрику RMSE (метрика, используемая для измерения точности модели, представляет собой квадратичный корень из средней квадратичной ошибки (MSE), которая вычисляется как среднее значение квадратов разностей между прогнозируемыми значениями модели и истинными значениями целевой переменной):

CNN:



RNN:



Квадратичный корень из средней квадратичной ошибки меньше у модели RNN, следовательно, она лучше всего подходит для нашей задачи прогнозирования цен на индексы и товары на финансовом рынке.

**ВЫВОДЫ**

В данной работе были проведены исследования и анализ финансового рынка с целью прогнозирования его дальнейшего развития. Были применены различные методы и модели, такие как временные ряды, статистические модели и машинное обучение, для прогнозирования будущих трендов и изменений на рынке.

Результаты исследования показали, что лучше всего себя показал модель RNN при прогнозировании цен на финансовые индексы и сырьевые товары.

Однако, следует отметить, что прогнозирование финансового рынка является сложной задачей, связанной с большой степенью неопределенности и риска. Влияние различных факторов, таких как политическая ситуация, экономические изменения и мировые события, может существенно влиять на динамику рынка и сделать точные прогнозы невозможными.

Таким образом, прогнозирование финансового рынка является важным инструментом для принятия решений в инвестиционной деятельности, однако необходимо учитывать его ограничения и принимать решения на основе комплексного анализа, включая не только прогнозы, но и другие факторы и индикаторы.

**Список литературы (использованных источников) и интернет-ресурсов**

1. Материал подготовлен Департаментом финансовой стабильности Центрального банка Российской Федерации. ИСПОЛЬЗОВАНИЕ БОЛЬШИХ ДАННЫХ В ФИНАНСОВОМ СЕКТОРЕ И РИСКИ ФИНАНСОВОЙ СТАБИЛЬНОСТИ [<https://www.cbr.ru/Content/Document/File/131359/Consultation_Paper_10122021.pdf>] Москва 2021.
2. Булгаков Андрей Леонидович. Big Data в финансах. Москва 2017. <https://cyberleninka.ru/article/n/big-data-v-finansah/viewer>
3. Kaggle [Электронный ресурс]. – Режим доступа: <https://www.kaggle.com/>, свободный.
4. Yahoo Finance [Электронный ресурс]. – Режим доступа: <https://finance.yahoo.com/>, свободный.
5. Investing.com [Электронный ресурс]. – Режим доступа: <https://www.investing.com/>, свободный.
6. Python Software Foundation. Python Language Reference, version 3.9.2 [Электронный ресурс]. – 2021. – Режим доступа: <https://docs.python.org/3/reference/>, свободный.

**ПРИЛОЖЕНИЕ**

Код программы для курсовой работы написано на языке Python

**Характеристики компьютера**

Процессор: AMD Ryzen 5 3600 6-Core Processor 3.95 GHz

Оперативная память: 16 гб

**Программный код**

Импорт библиотек:

import pandas as pd

from datetime import datetime, timedelta

import matplotlib.pyplot as plt

import warnings

from statsmodels.tsa.stattools import adfuller

import statsmodels.api as sm

from statsmodels.tsa.holtwinters import ExponentialSmoothing

from statsmodels.tsa.seasonal import STL

import itertools

from math import sqrt

from sklearn.metrics import mean\_squared\_error

import torch

import torch.nn as nn

from torch.utils.data import TensorDataset, DataLoader

from sklearn.model\_selection import train\_test\_split

from tqdm import tqdm

import torch.nn.functional as F

import torch.optim as optim

from torch.utils.data import DataLoader, TensorDataset

import numpy as np

warnings.filterwarnings("ignore")

Датасет включает в себя финансовые индексы и сырьевые товары

df = pd.read\_csv('Dataset\_DiMexRank\_36FinancialTimeSeries.csv')

df

df.dtypes

Приведём столбец 'Date' к формату даты и сделаем его индексом датафрейма

df['Date'] = pd.to\_datetime(df['Date'])

df.set\_index('Date', inplace=True)

df.isnull().sum()

df

# Проверка на стационарность с помощью Дики-Фуллера и скользящей средней

for column in df.columns:

result = adfuller(df[column])

print(f'ADF Statistic for {column}: {result[0]}')

print(f'p-value: {result[1]}')

print('Critical Values:')

for key, value in result[4].items():

print(f' {key}: {value}')

if result[1] < 0.05:

print('Ряд стационарен')

else:

print('Ряд нестационарен')

stationary\_mean = df[column].rolling(window=12).mean()

stationary\_std = df[column].rolling(window=12).std()

plt.figure(figsize=(10, 4))

plt.plot(df.index, df[column], label='Временной ряд')

plt.plot(df.index, stationary\_mean, color='red', linestyle='--', label='Скользящее среднее')

plt.fill\_between(df.index, stationary\_mean - stationary\_std, stationary\_mean + stationary\_std, color='gray', alpha=0.3, label='Скользящая вариация')

plt.xlabel('Дата')

plt.ylabel(column)

plt.title(f'Временной ряд {column}')

plt.legend()

plt.show()

# Тренд, сезонность и шум

for column in df.columns:

decomposition = sm.tsa.seasonal\_decompose(df[column], freq=40)

fig = decomposition.plot()

fig.set\_size\_inches(10, 8)

plt.suptitle(f"Декомпозиция временного ряда: {column}")

plt.show()

# Двойное дифференцирование данных

diff\_data = df.diff().diff(40).dropna()

for column in diff\_data.columns:

decomposition = sm.tsa.seasonal\_decompose(diff\_data[column], freq=40)

fig = decomposition.plot()

fig.set\_size\_inches(10, 8)

plt.suptitle(f"Декомпозиция временного ряда: {column}")

plt.show()

# Экспоненциальное сглаживание

df\_exp = df.copy()

for column in df\_exp.columns:

model = ExponentialSmoothing(df\_exp[column], seasonal\_periods=12, trend='add', seasonal='add')

result = model.fit()

df\_exp[column] = result.fittedvalues

plt.figure(figsize=(10, 4))

plt.plot(df\_exp.index, df\_exp[column], label='Seasonality Adjusted')

plt.xlabel('Date')

plt.ylabel(column)

plt.title('Seasonality Adjustment: ' + column)

plt.legend()

plt.show()

for column in df\_exp.columns:

decomposition = sm.tsa.seasonal\_decompose(df\_exp[column], freq=40)

fig = decomposition.plot()

fig.set\_size\_inches(10, 8)

plt.suptitle(f"Декомпозиция временного ряда: {column}")

plt.show()

# STL

df\_stl = df.copy()

for column in df\_stl.columns:

stl = STL(df\_stl[column])

result = stl.fit()

df\_stl[column] = result.resid

plt.figure(figsize=(10, 4))

plt.plot(df\_stl.index, df\_stl[column], label='Seasonality Adjusted')

plt.xlabel('Date')

plt.ylabel(column)

plt.title('Seasonality Adjustment: ' + column)

plt.legend()

plt.show()

for column in df\_stl.columns:

decomposition = sm.tsa.seasonal\_decompose(df\_stl[column], freq=40)

fig = decomposition.plot()

fig.set\_size\_inches(10, 8)

plt.suptitle(f"Декомпозиция временного ряда: {column}")

plt.show()

# SARIMA

for column in diff\_data.columns:

series = diff\_data[column]

train\_size = int(len(series) \* 0.9)

train\_data, test\_data = series[:train\_size], series[train\_size:]

p = d = q = range(0, 1)

pdq = list(itertools.product(p, d, q))

seasonal\_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))]

best\_aic = np.inf

best\_order = None

best\_seasonal\_order = None

for param in pdq:

for param\_seasonal in seasonal\_pdq:

try:

model = sm.tsa.statespace.SARIMAX(train\_data, order=param, seasonal\_order=param\_seasonal)

results = model.fit()

if results.aic < best\_aic:

best\_aic = results.aic

best\_order = param

best\_seasonal\_order = param\_seasonal

except:

continue

best\_model = sm.tsa.statespace.SARIMAX(train\_data, order=best\_order, seasonal\_order=best\_seasonal\_order)

best\_results = best\_model.fit()

forecast = best\_results.get\_forecast(steps=len(test\_data))

forecasted\_values = forecast.predicted\_mean

plt.figure(figsize=(10, 6))

plt.plot(series.index, series, label='Actual')

plt.plot(test\_data.index, forecasted\_values, linestyle='--', marker='o', label='Forecast')

plt.fill\_between(test\_data.index, forecast.conf\_int().iloc[:, 0], forecast.conf\_int().iloc[:, 1], color='gray', alpha=0.3)

plt.title(f'SARIMA Forecast for {column}')

plt.xlabel('Date')

plt.ylabel('Value')

plt.legend()

rmse = sqrt(mean\_squared\_error(test\_data, forecasted\_values))

plt.text(series.index[-1], series.iloc[-1], f'RMSE: {rmse:.2f}', ha='right', va='top')

plt.show()

diff\_data

# CNN

import numpy as np

import matplotlib.pyplot as plt

import torch

import torch.nn as nn

from torch.utils.data import TensorDataset, DataLoader

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error

from tqdm import tqdm

import torch.nn.functional as F

def split\_sequences(sequences):

X, y = [], []

for i in range(len(sequences)):

end\_ix = i + 1

if end\_ix > len(sequences) - 1:

break

seq\_x, seq\_y = sequences[i:end\_ix], sequences[end\_ix]

X.append(seq\_x)

y.append(seq\_y)

return np.array(X), np.array(y)

dataset = diff\_data.to\_numpy()

n\_features = dataset.shape[1]

X\_train\_list = []

X\_test\_list = []

y\_train\_list = []

y\_test\_list = []

for i in range(n\_features):

X\_i, y\_i = split\_sequences(dataset[:, i])

X\_train\_i, X\_test\_i, y\_train\_i, y\_test\_i = train\_test\_split(X\_i, y\_i, test\_size=0.1, random\_state=42)

X\_train\_list.append(X\_train\_i)

X\_test\_list.append(X\_test\_i)

y\_train\_list.append(y\_train\_i)

y\_test\_list.append(y\_test\_i)

X\_train\_tensor = torch.Tensor(np.stack(X\_train\_list, axis=1))

y\_train\_tensor = torch.Tensor(np.stack(y\_train\_list, axis=1))

X\_test\_tensor = torch.Tensor(np.stack(X\_test\_list, axis=1))

y\_test\_tensor = torch.Tensor(np.stack(y\_test\_list, axis=1))

class ConvNet(nn.Module):

def \_\_init\_\_(self, n\_channels, n\_features):

super(ConvNet, self).\_\_init\_\_()

self.conv1 = nn.Conv1d(n\_channels, 64, kernel\_size=1)

self.pool = nn.MaxPool1d(kernel\_size=2, stride=1, padding=1)

self.flatten = nn.Flatten()

self.fc1 = nn.Linear(128, 50)

self.fc2 = nn.Linear(50, n\_features)

def forward(self, x):

x = self.conv1(x)

x = self.pool(x)

x = self.flatten(x)

x = F.relu(self.fc1(x))

x = self.fc2(x)

return x

n\_channels = X\_train\_tensor.shape[1]

model = ConvNet(n\_channels, n\_features)

loss\_fn = nn.MSELoss()

optimizer = torch.optim.Adam(model.parameters(), lr=0.001)

train\_dataset = TensorDataset(X\_train\_tensor, y\_train\_tensor)

train\_loader = DataLoader(train\_dataset, batch\_size=32, shuffle=True)

epochs = 100

progress\_bar = tqdm(total=epochs, desc='Training Progress')

for epoch in range(epochs):

model.train()

for batch\_X, batch\_y in train\_loader:

pred\_y = model(batch\_X)

loss = loss\_fn(pred\_y, batch\_y)

optimizer.zero\_grad()

loss.backward()

optimizer.step()

progress\_bar.update(1)

progress\_bar.close()

model.eval()

y\_pred = model(X\_test\_tensor)

y\_test\_pred = y\_pred.detach().numpy()

y\_test\_np = y\_test\_tensor.numpy()

rmse\_values = []

for i in range(n\_features):

y\_pred\_i = y\_test\_pred[:, i]

y\_test\_i = y\_test\_np[:, i]

rmse = np.sqrt(mean\_squared\_error(y\_test\_i, y\_pred\_i))

rmse\_values.append(rmse)

x = np.arange(len(dataset))

plt.figure(figsize=(10, 6))

plt.plot(x, dataset[:, i], label='Actual', color='blue')

plt.plot(x[-len(y\_pred):], y\_pred.detach().numpy()[:, i], label='Predicted', color='red')

plt.xlabel('Time')

plt.ylabel('Value')

plt.title(f'Actual vs Predicted (Feature {i+1})')

plt.legend()

plt.text(0.02, 0.85, f'RMSE: {rmse:.4f}', transform=plt.gca().transAxes)

plt.show()

print(f'RMSE: {rmse:.4f}')

plt.figure(figsize=(8, 4))

plt.bar(range(n\_features), rmse\_values)

plt.xlabel('Feature')

plt.ylabel('RMSE')

plt.title('RMSE for Each Feature')

plt.show()

# RNN

import numpy as np

import matplotlib.pyplot as plt

import torch

import torch.nn as nn

from torch.utils.data import TensorDataset, DataLoader

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error

from tqdm import tqdm

def split\_sequences(sequences):

X, y = [], []

for i in range(len(sequences)):

end\_ix = i + 1

if end\_ix > len(sequences) - 1:

break

seq\_x, seq\_y = sequences[i:end\_ix], sequences[end\_ix]

X.append(seq\_x)

y.append(seq\_y)

return np.array(X), np.array(y)

dataset = diff\_data.to\_numpy()

n\_features = dataset.shape[1]

X\_train\_list = []

X\_test\_list = []

y\_train\_list = []

y\_test\_list = []

for i in range(n\_features):

X\_i, y\_i = split\_sequences(dataset[:, i])

X\_train\_i, X\_test\_i, y\_train\_i, y\_test\_i = train\_test\_split(X\_i, y\_i, test\_size=0.1, random\_state=42)

X\_train\_list.append(X\_train\_i)

X\_test\_list.append(X\_test\_i)

y\_train\_list.append(y\_train\_i)

y\_test\_list.append(y\_test\_i)

X\_train\_tensor = torch.Tensor(np.stack(X\_train\_list, axis=1))

y\_train\_tensor = torch.Tensor(np.stack(y\_train\_list, axis=1))

X\_test\_tensor = torch.Tensor(np.stack(X\_test\_list, axis=1))

y\_test\_tensor = torch.Tensor(np.stack(y\_test\_list, axis=1))

class RNN(nn.Module):

def \_\_init\_\_(self, input\_size, hidden\_size, output\_size):

super(RNN, self).\_\_init\_\_()

self.hidden\_size = hidden\_size

self.rnn = nn.RNN(input\_size, hidden\_size, batch\_first=True)

self.fc = nn.Linear(hidden\_size, output\_size)

def forward(self, x):

hidden = torch.zeros(1, x.size(0), self.hidden\_size).to(x.device)

out, \_ = self.rnn(x, hidden)

out = self.fc(out[:, -1, :])

return out

input\_size = X\_train\_tensor.shape[2]

hidden\_size = 64

output\_size = y\_train\_tensor.shape[1]

model = RNN(input\_size, hidden\_size, output\_size)

loss\_fn = nn.MSELoss()

optimizer = torch.optim.Adam(model.parameters(), lr=0.001)

train\_dataset = TensorDataset(X\_train\_tensor, y\_train\_tensor)

train\_loader = DataLoader(train\_dataset, batch\_size=32, shuffle=True)

epochs = 100

progress\_bar = tqdm(total=epochs, desc='Training Progress')

for epoch in range(epochs):

model.train()

for batch\_X, batch\_y in train\_loader:

pred\_y = model(batch\_X)

loss = loss\_fn(pred\_y, batch\_y)

optimizer.zero\_grad()

loss.backward()

optimizer.step()

progress\_bar.update(1)

progress\_bar.close()

model.eval()

y\_pred = model(X\_test\_tensor)

y\_test\_pred = y\_pred.detach().numpy()

y\_test\_np = y\_test\_tensor.numpy()

rmse\_values = []

for i in range(n\_features):

y\_pred\_i = y\_test\_pred[:, i]

y\_test\_i = y\_test\_np[:, i]

rmse = np.sqrt(mean\_squared\_error(y\_test\_i, y\_pred\_i))

rmse\_values.append(rmse)

x = np.arange(len(dataset))

plt.figure(figsize=(10, 6))

plt.plot(x, dataset[:, i], label='Actual', color='blue')

plt.plot(x[-len(y\_pred):], y\_pred.detach().numpy()[:, i], label='Predicted', color='red')

plt.xlabel('Time')

plt.ylabel('Value')

plt.title(f'Actual vs Predicted (Feature {i+1})')

plt.legend()

plt.text(0.02, 0.85, f'RMSE: {rmse:.4f}', transform=plt.gca().transAxes)

plt.show()

print(f'RMSE: {rmse:.4f}')

plt.figure(figsize=(8, 4))

plt.bar(range(n\_features), rmse\_values)

plt.xlabel('Feature')

plt.ylabel('RMSE')

plt.title('RMSE for Each Feature')

plt.show()