МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Факультет Компьютерных технологий и прикладной математики
Кафедра анализа данных и искусственного интеллекта

КУРСОВАЯ РАБОТА

ПРОГНОЗИРОВАНИЕ ФОНДОВОГО РЫНКА МЕТОДАМИ АНАЛИЗА ВРЕМЕННЫХ РЯДОВ

Работу выполнила _____ (подпись) В.А. Головань

Направление подготовки 09.03.03 Прикладная информатика

Направленность (профиль) Прикладная информатика в экономике

Научный руководитель
канд. физ.-мат. наук, доц. _____ (подпись) Е.В. Кособуцкая

Нормоконтролер
канд. физ.-мат. наук, доц. _____ (подпись) Г.В. Калайдина

Краснодар
2021

РЕФЕРАТ

Курсовая работа 33 страницы, 11 рисунков, 13 источников.

ПРОГНОЗИРОВАНИЕ ФОНДОВОГО РЫНКА МЕТОДАМИ АНАЛИЗА ВРЕМЕННЫХ РЯДОВ.

Объектом исследования является прогнозирование цены акций.

Предмет исследования – временные ряды и их особенности, фондовый рынок.

Актуальность данной курсовой работы обусловлена тем, что фондовый рынок является труднопредсказуемой областью экономики. Людям с долгосрочной инвестиционной стратегией необходимо понимать в какой момент следует продать акции, а в какой нет.

Целью курсовой является изучение языка программирования Python3, библиотек matplotlib, NumPy и Pandas, их возможностей для работы с временными рядами, а также различных способов анализа, моделирования и прогнозирования временных рядов. Разработать программную реализацию для сглаживания и оценивая временного ряда.

В ходе работы был произведен сравнительный анализ методов работы с временными рядами, моделей прогнозирования временных рядов, был изучен язык Python и его библиотеки.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Фондовый рынок отличается своей нестабильность. Каждый день котировки акций растут или снижаются. Выбрать лучшее время для покупки или продажи очень непросто, так как на фондовый рынок влияют состояние экономики, политика, настроение инвесторов и общее положение дел в мире.

Ключевые фондовые рынки всегда живут ожиданиями будущих событий. Серьезным экономическим спадам в отдельных странах и регионах обычно предшествует падение соответствующих рынков акций.

В современных реалиях в силу быстро изменяющихся процессах и неопределенности информации наибольшую популярность набирает краткосрочное прогнозирование. При таком прогнозировании наибольший приоритет имеют последние данные исследуемого процесса, а не сложившаяся тенденция.

Такие прогнозы позволяют не только оптимизировать прибыль, но и экономические и социальные катастрофы. Накопленная информация часто используется при разработке экономических теорий и финансовых приложений. Обработка исторических временных рядов экономических величин позволила сгладить очередные финансовые кризисы.

Целью курсовой является изучение языка программирования Python3, библиотек matplotlib, NumPy и Pandas, их возможностей для работы с временными рядами, а также различных способов анализа, моделирования и прогнозирования временных рядов. Разработать программную реализацию для сглаживания и оценивая временного ряда.

1 Временные ряды

1.1 Основные понятия анализа временных рядов

Под временным (динамическим) рядом понимают последовательность наблюдений некоторого признака X (случайной величины) в последовательные моменты времени t. Уровнями ряда называются отдельные наблюдения, которые обозначаются x_t, t= 1,..,n.

При исследовании временного ряда выделяют несколько составляющих:

$$x_t = u_t + \gamma_t + c_t + \varepsilon_t, t = 1 \dots n,$$

где u_t – тренд, плавно меняющаяся компонента, описывающая чистое влияние долговременны факторов,

γ_t– сезонная компонента, отражающая повторяемость процессов в течение короткого периода (день, неделя и так далее),

c_t – циклическая компонента, отражающая повторяемость процессов в течение длительных периодов времени свыше одного года,

ε_t – случайная компонента, отражающая влияние неподдающихся учету и регистрации случайных факторов.

Временной ряд рассматривается, как одна из реализаций случайного процесса X(t)(t – время), а его члены, как правило, не являются статистически независимыми и одинаково распределенными. [12]

Временные ряды описываются по следующим признакам:

– по форме представления уровней: ряды абсолютных показателей, относительных показателей, средних величин;

– по количеству показателей, для которых указаны уровни в определенный момент времени: одномерные и многомерные временные ряды;

– по характеру временного параметра: моментные и интервальные временные ряды;

– по расстоянию между датами и интервалами времени: равностоящие, то есть, когда даты начала и окончания периодов следуют друг за другом с одинаковыми интервалами, неполные – интервалы разные.

– по полноте данных: полные и неполные;

– детерминированные и случайные. Первые получаются на основе значений некоторой неслучайной функции, вторые – результат реализации некоторой случайной величины;

– по наличию основной тенденции: стационарные и нестационарные ряды.

Можно выделить следующие этапы анализа и прогнозирования временных рядов:

– графическое представление и анализ составляющих ряда;

– проверка ряда на стационарность;

– сглаживание и фильтрация (удаление низко- или высокочастотных составляющих) временного ряда;

– построение модели прогнозирования;

– прогнозирование временного ряда на основе ранее проведенных исследований. [4]

Важный этап, с которого начинается сам анализ и прогнозирование, заключается в поиске данных и их подготовке.

Анализ временного ряда прежде всего начинается с поиска данных и приведения их к виду, доступному для прогнозирования. Необходимо определить является ли исследуемый временной ряд стационарным или нет.

Временной ряд x_t, t=1,..,n называется строго стационарным, если совместное распределение вероятностей n наблюдений x_1, x_2,…, $x_{n+\tau}$ при любых n, t, τ. То есть у стационарных временных рядов вероятностные характеристики не зависят от момента t. Поэтому математическое ожидание $M(x_t)=a$ и среднеквадратическое отклонение могут быть оценены по значениям x_t по формулам:

$$a = \bar{x}_t = \sum_{t=1}^{n} x_t / n,$$

$$\sigma = \sqrt{\frac{\sum_{t=1}^{n} (x_t - \bar{x}_t)^2}{n}}.$$

Из стационарности временного ряда следует, что такие свойства, как дисперсия, математическое ожидание и ковариация неизменны со временем. [5]

Для того, чтобы проверить временной ряд на стационарность, необходимо провести тест Дикки-Фуллера, который проверяет нулевую гипотезу о наличии единичного корня.

Если из обеих частей уравнения $x_t = p \cdot x_t + \varepsilon_t$, отображающего наш процесс, вычесть x_{t-1}, то получим $x_t - x_{t-1} = (p-1) \cdot x_{t-1} + \varepsilon_t$, где слева от равенства – первые разности. Из этого следует, если $p = 1$, то первые разности дадут стационарный белый шум, то есть временной ряд будет невозможно предсказать. Этот факт и лег в основу теста Дики-Фуллера на наличие единичных корней. [3]

Нулевая гипотеза в критерии Дики-Фуллера состоит в том, что ряд не стационарен и имеет один единичный корень $p = 1$.

Альтернативная – в том, что ряд стационарен и $p < 1$. Для получения статистики, с помощью которой можно было бы проверить нулевую гипотезу, Дики и Фуллер предложили оценить данную авторегрессионную модель и взять из неё обычную t-статистику для гипотезы о том, что $p = 1$.

Предположение о том, что переменная следует авторегрессионному процессу первого порядка при условии некоррелируемых ошибок является несколько ограничительным. Дополненный критерий Дики-Фуллера (augmented Dickey-Fuller test, ADF) был модифицирован для авторегрессионных процессов более высоких порядков. Уравнение для него приобретает следующий вид:

$$\Delta x_t = (p-1)x_{t-1} + \sum_{j=1}^{k} x_j \Delta x_{t-j} + \varepsilon_t,$$

$$\Delta x_t = (p-1)x_{t-1} + \varepsilon_t.$$

Распределения этих критериев асимптотически совпадают с соответствующими обычными распределениями Дики-Фуллера. Роль дополнительной авторегрессионной компоненты сводится к тому, чтобы убрать автокорреляцию из остатков. Процедура проверки гипотез не отличается от описанной выше.

Таким образом, если $p-\text{значение} > 0.05$, то не удается отклонить нулевую гипотезу, следовательно существует единичный корень и ряд является нестационарным.

Если же $p-\text{значение} < 0.05$, то нулевая гипотеза отклоняется, единичного корня не существует и ряд является стационарным.

Следующим этапом прогнозировании временного ряда является сглаживание.

1.2 Методы сглаживания временных рядов

1.2.1 Скользящая средняя

Выделяют несколько видов сглаживания (выравнивания) временных рядов, для приведения к стационарности.

Скользящая средняя (Moving Average, MA) – общее название для семейства функций, значения которых в каждой точке определения равны среднему значению исходной функции за предыдущий период.

MA можно строить разными способами:

Базовая MA – простое или арифметическое скользящее среднее (Simple Moving Average, SMA). SMA в момент времени t (SMAt) рассчитывается, как среднее арифметическое n значений ряда:

$$SMA_t = \frac{1}{n} \cdot \sum_{i=0}^{n-1} P_{t-i},$$

где SMA_t – значение простого скользящего среднего в точке t;

n – количество значений исходной функции для расчёта скользящего среднего, чем шире сглаживающий интервал, тем более плавным получается график функции;

P_{t-i} – значение исходной функции в точке t-i.

Одним из плюсов данного метода является его простота и ясность, но есть SMA_t ряд недостатков:

– все значения входят с одинаковым весом;

– двойное реагирование на слагаемое при перемещении от окна к окну;

– в первой точке и последней точке нельзя рассчитать сглаженные значения, так как не существует соответственно прошлого и будущего значений по отношению к рассчитываемому.

В случае, когда выбросы имеют редкий и случайный характер, для выделения регулярной составляющей метод, скользящего среднего неприемлем.

Вместе с тем, для временного ряда типична ситуация, когда одни его отсчеты более значимы, другие менее. Для корректного сглаживания такого временного используют взвешенную скользящую среднюю (Weighted Moving Average, WMA).

WMA в момент t (WMAt):

$$WMA_t = \sum_{i=0}^{n-1} w_{t-i} \cdot p_{t-i} \,,$$

где p_{t-i} − значение ряда в точке (момент времени) t-i;

w_{t-i} − нормированный вес уровня p_{t-i}.

Нормированные веса удовлетворяют традиционному условию:

$$\sum_{i=0}^{n-1} w_{t-i} = 1,$$

То есть, их сумма по данному интервалу (окну) равна одному.

1.2.2 Экспоненциальное сглаживание

В отличие от метода скользящих средних, экспоненциальное сглаживание может применяться для краткосрочных прогнозов будущей тенденции на один период вперед. Именно поэтому оно обладает явным преимуществом над ранее рассмотренными методами.

Экспоненциальное сглаживание можно представить как фильтр, на вход которого последовательно поступают члены исходного ряда, а на выходе формируются текущие значения экспоненциальной средней.

$$\mathrm{E}_i = a\Upsilon_i + (1-a)\cdot \mathrm{E}_{i-1},$$

где Υ_i – наблюдаемое значение в данной точке ряда,

E_{i-1} – рассчитанное глаженное значение для предшествующей точки ряда,

a – некоторый заранее заданный коэффициент сглаживания, постоянный по всему ряду, $0 < a < 1$.

Выбор коэффициента сглаживания в значительной степени влияет на результаты – чем он больше (ближе к 1), тем больше вес последних значений и тем меньше вес предыдущих.

На основе простого экспоненциального сглаживания были разработаны более сложные модели сглаживания временных рядов, содержащих периодические сезонные колебания и/или обладающих тенденцией роста.

Данная система позволяет строить наряду с простым экспоненциальным сглаживанием модели, отражающие эффекты роста (линейного, экспоненциального или затухающего) и сезонности (аддитивного или мультипликативного), которыми обладает исходный ряд.

При экспоненциальном сглаживании не учитываются тренд и сезонные колебания, что является недостатком.

1.2.3 Двойное экспоненциальное сглаживание

В отличие от предыдущих методов, двойное экспоненциальное сглаживание делает прогноз не на одну, а на две точки вперед и так же позволяет сглаживать ряд.

Для этого временной ряд разбивается на 2 составляющие:

– уровень l;

– тренд b – зависит от рассчитанного сглаженного значения за предыдущий и текущий периоды.

Применим экспоненциальное сглаживание, полагая, что будущее направление изменения ряда зависит от взвешенных предыдущих значений:

$$l_x = \alpha y_x + (1 - \alpha)(l_{x-1} + b_{x-1}) ,$$

$$b_x = \beta(l_x + l_{x-1}) + (1 - \beta)b_{x-1} ,$$

$$\hat{y}_{x+1} = l_x + b_x .$$

В результате получаем набор функций. Первая описывает уровень, зависящий от текущего значения ряда, а второе слагаемое теперь разбивается на предыдущее значение уровня и тренда. Вторая отвечает за тренд, который зависит от изменения уровня на текущем шаге, и от предыдущего значения тренда. Здесь в роли веса в экспоненциальном сглаживании выступает коэффициент β. Наконец, итоговое предсказание представляет собой сумму модельных значений уровня и тренда.

2 Методы для построения моделей временных рядов

После приведения ряда к стационарному, можно приступать к построению модели. Существует несколько видов моделей. Наиболее популярными являются интегрированная модель авторегрессии ARIMA, рекуррентная нейронная сеть LSTM и регрессионные модели прогнозирования. На практике можно убедиться, что не существует универсального метода для прогнозирования, так как надо учитывать характерные особенности каждого временного ряда. [1]

2.1 Сезонная ARIMA модель SARIMA

Сезонная авторегрессионная интегрированная модель SARIMA или Seasonal ARIMA, является расширением ARIMA (p, d, q), которое поддерживает одномерные данные временных рядов с сезонным компонентом.

$$\Delta^d Y_t = \sum_{i=1}^{p} \alpha_i \Delta^d Y_{t-1} + \sum_{j=1}^{q} \beta_j \varepsilon_{t-j} + \varepsilon_t,$$

где ε_t – стационарный временной ряд,

α_i, β_j – параметры модели,

Δ^d – оператор разности d-го порядка (последовательное взятие d раз разностей первого порядка — сначала от временного ряда, затем от полученных разностей первого порядка, затем от второго порядка и т. д.)

Модель SARIMA (p, d, q)(P, D, Q)s – обобщение ARIMA-модели на временные ряды, в которых имеется ярко выраженная сезонная компонента. Дополнительно в такой модели вводятся сезонные параметры (P, D, Q, s), позволяющие учесть циклические колебания процесса, где P – порядок сезонной составляющей SAR(P), D – порядок интегрирования сезонной составляющей, Q – порядок сезонной составляющей SMA(Q), s— размерность сезонности (месяц, квартал и т.д.).

Несмотря на свою популярность, модель SARIMA является сложной в реализации, так как дополнительно требуется определять сезонные параметры. В языке Python3 нет отдельного метода для построения данной модели, но можно задать функцию для нахождения параметров перебором. В этом случае необходимо сохранять наилучшую модель для дальнейшего обучения.

Данная модель может не подойти для некоторых проектов, так как использует большое количество ресурсов, в том числе времени, на первоначальных этапах подготовки, сложно настраивается, а также требует переобучения на новых данных. [9]

2.2 LSTM

LSTM – рекуррентная нейронная сеть с долгой кратковременной памятью. С ее помощью есть возможность выявлять признаки из временных последовательностей, а также обрабатывать многомерные данные.

При обучении данная модель способна схватывать существенные детали прошлого контекста и сохранять их, пока они актуальны.

В базовой версии LSTM состоит из ячеек. Все рекуррентные нейронные сети имеют форму цепочки повторяющихся модулей нейронной сети. В стандартных РНС этот повторяющийся модуль имеет простую структуру, например, один слой *tanh*.

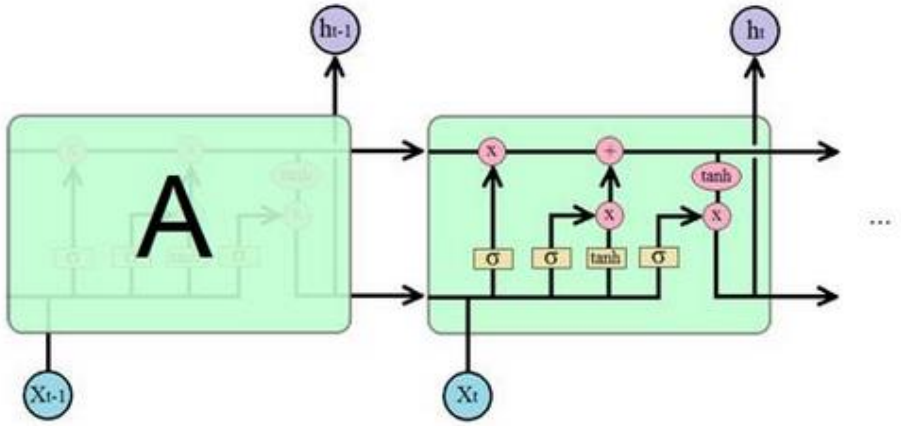

Рисунок 1 – Схема работы LSTM модели

На приведенной выше диаграмме каждая линия является вектором. Розовый круг означает поточечные операции, например суммирование векторов. Под желтыми ячейками понимаются слои нейронной сети.

σ – сигмовидный слой модели (гейт), пропускающий или не пропускающий информацию. Состоит из операции поточечного умножения, в результате чего на выходе сигмовидного слоя выдаются числа от нуля до единицы определяя, сколько процентов каждой единицы информации пропустить дальше. Значение «0» означает «не пропустить ничего», значение «1» – «пропустить все». В LSTM существует три гейта для контроля за состоянием ячейки.

Схема работы модели состоит из трёх этапов: слой утраты, слой сохранения и новое состояние.

На первом этапе LSTM нужно решить, какую информацию мы собираемся выбросить из состояния ячейки. Это решение принимается сигмовидным слоем, называемым «слоем гейта утраты». Он получает на вход h и x и выдает число от 0 до 1 для каждого номера в состоянии ячейки C. Единица означает «полностью сохранить», а ноль — «полностью удалить».

$$f_t = \sigma(W_t \cdot [h_{t-1}, x_t] + b_f),$$

Второй этап отвечает за сохранение информации в ячейке. Процесс разбивается на две части, где сначала сигмовидный слой решает какое значение требуется обновить, затем слой *tanh* создает вектор новых значений кандидатов *C*, которые добавляются в состояние. На следующем шаге мы объединим эти два значения для обновления состояния.

$$i_t = \sigma(W_t \cdot [h_{t-1}, x_t] + b_i),$$

$$\tilde{C}_t = tanh\sigma(W_C \cdot [h_{t-1}, x_t] + b_C).$$

Завершающим этапом является обновление предыдущего состояния ячейки для получения нового состояния *C*. Умножим старое состояние на f, теряя информацию, которую решили забыть. Затем добавляем i·C. Это новые значения кандидатов, масштабируемые в зависимости от того, как мы решили обновить каждое значение состояния.

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t.$$

Наконец, нужно решить, что хотим получить на выходе. Результатом будет являться отфильтрованным состоянием ячейки. Сначала запускаем сигмовидный слой, который решает, какие части состояния ячейки выводить. Затем пропускаем состояние ячейки через tanh (чтобы разместить все значения в интервале [-1, 1]) и умножаем его на выходной сигнал сигмовидного гейта.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o),$$

$$h_t = o_t \cdot tan\, h(C_t).$$

К достоинствам модели LSTM можно отнести способность эффективно строить нелинейные зависимости, точно описывающие исходные данные, параллелизм и адаптивность. [2]

2.3 Регрессионная модель

Прогнозируемая переменная зависит от нескольких независимых переменных.

Выделяют 3 типа регрессионной модели прогнозирования:

– простая линейная регрессия,

– множественная регрессия,

– нелинейная регрессия;

В основу простой линейной регрессии положено предположение, что существует дискретный внешний фактор *X(t)*, оказывающий влияние на исследуемый процесс *Z(t)*, при этом связь между процессом и внешним фактором линейна.

$$Z(t) = \alpha_0 + \alpha_1 X(t) + \varepsilon_t,$$

где α_0 и α_1 – коэффициенты регрессии;

ε_t – ошибка модели.

Недостатком такого способа прогнозирования является то, что для получения значений *Z(t)* в момент времени *t* необходимо иметь значение *X(t)* в тот же момент времени *t*, что редко выполнимо на практике.

В действительности на процесс *Z(t)* оказывают влияние множество дискретных внешних факторов $X_1(t) \dots X_s(t)$, тогда применяется множественная регрессия, которая имеет вид:

$$Z(t) = \alpha_0 + \alpha_1 X_1(t) + \alpha_2 X_2(t) + \cdots + \alpha_s X_s(t) + \varepsilon_t.$$

В основу нелинейной регрессионной модели положено предположение о том, что существует известная функция, описывающая зависимость между исходным процессом *Z(t)* и внешним фактором *X(t).*

$$Z(t) = F(X(t), A).$$

Основной недостаток регрессионных моделей является наличие заранее известных значений всех факторов. [8]

3 Прогнозирование фондового рынка методами анализа временных рядов

3.1 Постановка задачи

Для анализа и прогнозирования фондового рынка нам необходимы данные с ценами на акции за определенный период времени.

Для рассматриваемой компании Amazon эти данные можно скачать с сайта Yahoo.finance в период с 15.11.2004 по 12.11.2021. Размер файла представляет собой набор из 3526 строк и 7 колонок. Столбец Date содержит информацию о днях, Open показывает цену акции на момент открытия фондовой биржи, Close – закрытия. High и Low отображают максимальный и минимальный уровень цены во время торгов, Adj Close – скорректированная стоимость на момент закрытия, Volume показывает количество сделок в данном временном периоде. [7]

Как правило, все сделки оцениваются по цене закрытия, поэтому нам понадобится только столбец Close.

В ходе анализа и дальнейшего прогнозирования, будут строиться различные графики, чтобы продемонстрировать поведение исследуемого временного ряда.

Результаты прогнозов должны быть представлены в графическом виде.

В основе моделирования применяется SARIMA, так как ряд имеет тренд и сезонность.

3.2 Программная реализация

Для работы с данными и построения прогноза на цены, будем использовать Python3. При помощи встроенных библиотек реализуются математические и статистические методы.

В основном, нам понадобятся 3 библиотеки и их методы:

– библиотека Pandas для работы с наборами данных;

– библиотека NumPy для работы с математическими операциями;

– библиотека Matplotlib для построения графиков.

Выбор среды разработки JupyterNotebook обусловлен его простотой в работе написания кода, раздельным компилированием и возможностью отображения графиков без использования консоли.

Для начала работы импортируем наши данные из файла в среду JupyterNotebook и выведем их на экран.

С помощью метода read_csv() библиотеки pandas мы можем прочитать данные из ранее загруженного файла, чтобы посмотреть, как они представлены, какие имеют типы и производить дальнейшую работу. Если же в датафрейме присутствуют нулевые значения или столбцы, которые не несут значимой информации, датасет необходимо скорректировать, убрав эти значения.

```
a=read_csv('AMZN.csv')
a
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2004-11-15	40.459999	41.500000	40.180000	40.889999	40.889999	10366200
1	2004-11-16	40.500000	40.660000	39.750000	40.070000	40.070000	6401300
2	2004-11-17	40.400002	40.590000	39.540001	39.900002	39.900002	6843600
3	2004-11-18	39.730000	40.959999	39.590000	40.369999	40.369999	6806800
4	2004-11-19	39.150002	39.439999	38.189999	38.549999	38.549999	11427500
...	...	...	...	...	...	...	...
4275	2021-11-08	3523.239990	3579.000000	3487.860107	3488.979980	3488.979980	3074000
4276	2021-11-09	3515.250000	3593.770020	3501.429932	3576.229980	3576.229980	4294900
4277	2021-11-10	3563.870117	3605.449951	3463.090088	3482.050049	3482.050049	4027400
4278	2021-11-11	3513.000000	3543.239990	3467.469971	3472.500000	3472.500000	2264400
4279	2021-11-12	3485.000000	3540.729980	3447.050049	3525.149902	3525.149902	2688500

4280 rows × 7 columns

Рисунок 2 – Вывод данных на экран

Можно убедиться, что данные читаются из файла и отображаются для дальнейшей работы.

Как говорилось ранее, котировки акций смотрят по цене на момент закрытия фондовой биржи, поэтому из всего датафрейма будем работать со столбцом amazon.Close. Для удобства выделим в отдельный набор данных и отобразим их.

```
amazonClose = amazon.Close
amazonClose
```

```
0          29.500000
1          27.812500
2          27.437500
3          25.812500
4          24.250000
             ...
5277     3488.979980
5278     3576.229980
5279     3482.050049
5280     3472.500000
5281     3525.149902
Name: Close, Length: 5282, dtype: float64
```

Рисунок 3 – Выделение определенного столбца в отдельный датасет

Построим график нашего временного ряда, чтобы наглядно увидеться как изменялась цена акций с течением времени.

На рисунке представлен фрагмент кода, который служит для графического отображения цены акций с течением времени.

Функция plt.figure() является контейнером самого верхнего уровня и помогает определить область, в которой будет отображаться график.

Метод plt.plot() непосредственно строит график, в качестве параметра передаются значения для его построения.

С помощью plt.ylabel() и plt.xlabel() отображаем названия осей ординат и абцисс.

plt.grid() показывает линии сетки на графике, параметром False указано, что отображение сетки не требуется.

```
plt.figure(figsize=(15, 7))
plt.plot(amazonClose)
plt.ylabel('Closing price ($)')
plt.xlabel('Trading day')
plt.grid(False)
```

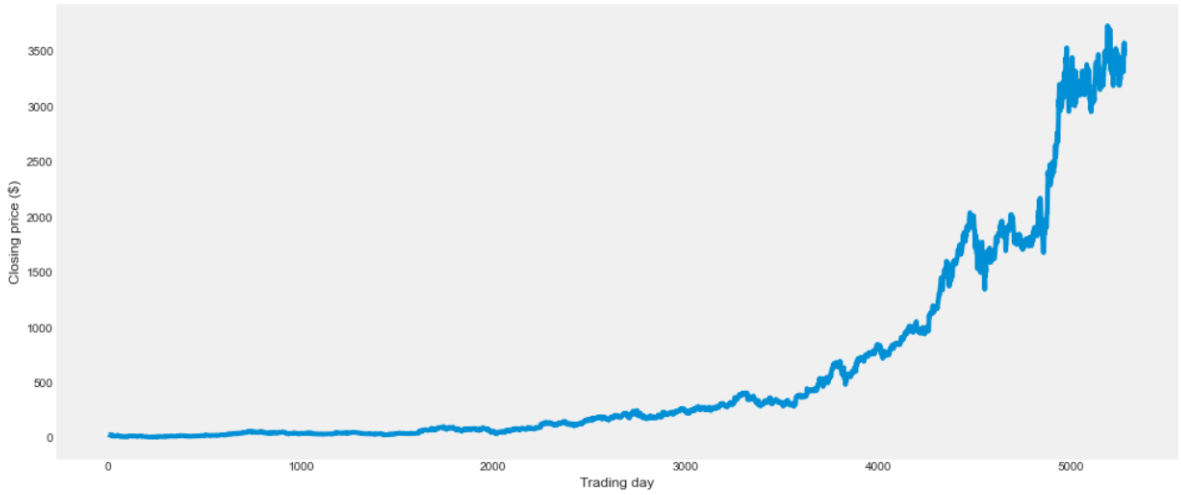

Рисунок 4 – Отображение цены акций в виде графика

Многие методы и модели основаны на предположениях о стационарности ряда (см. Глава 1), но как мы можем увидеть по графику, наш ряд таковым не является. Поэтому проведем обобщенный тест Дикки-Фуллера на наличие единичных корней.

Для этого в statsmodels есть функция adfuller().

Statsmodels – это модуль Python, который предоставляет классы и функции для оценки различных статистических моделей, а также для проведения статистических тестов и исследования статистических данных.

```
test = sm.tsa.adfuller(amazonClose)
print ('adf: ', test[0])
print ('p-value: ', test[1])
if test[0]> test[1]:
    print ('есть единичные корни, ряд не стационарен')
else:
    print ('единичных корней нет, ряд стационарен')
```

```
adf:  3.435216710422938
p-value:  1.0
есть единичные корни, ряд не стационарен
```

Рисунок 5 – Первый тест Дикки-Фуллера

Итак, тест подтвердил предположение о не стационарности временного ряда (см. Глава 1).

Сгладим набор данных, применив экспоненциальное сглаживание и еще раз проведем тест Дикки-Фуллера.

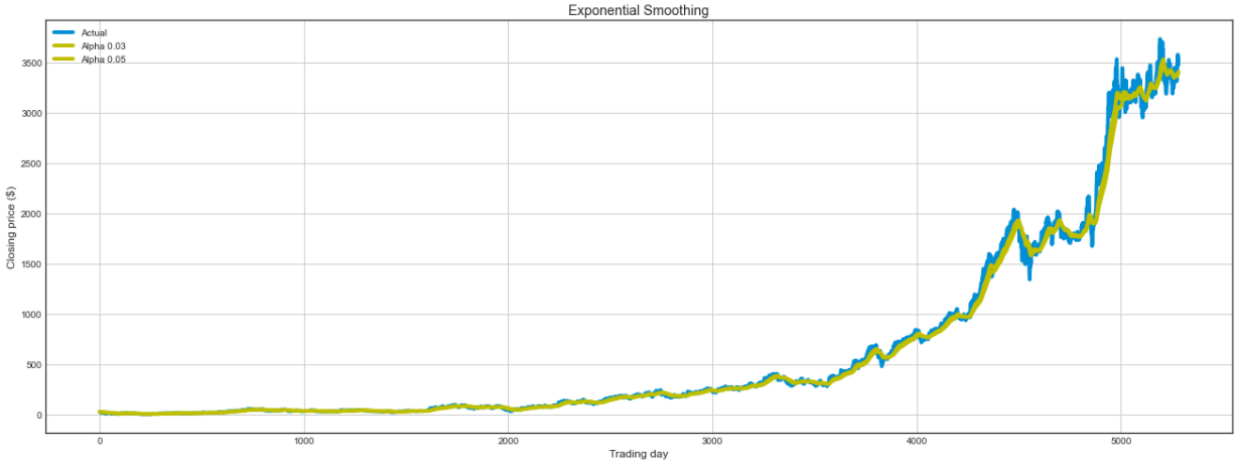

Рисунок 6 – График экспоненциального сглаживания

```
test = sm.tsa.adfuller(amazonClose)
print ('adf: ', test[0])
print ('p-value: ', test[1])
if test[0]> test[1]:
    print ('есть единичные корни, ряд не стационарен')
else:
    print ('единичных корней нет, ряд стационарен')
```

```
adf:  -14.095068394248413
p-value:  2.679387630802616e-26
единичных корней нет, ряд стационарен
```

Рисунок 7 – Повторный тест Дикки-Фуллера

По результатам повторного теста можно опровергнуть нулевую гипотезу о наличии единичного корня, следовательно верна альтернативная гипотеза. Наш ряд получился стационарным, его дисперсия, математическое ожидание и ковариация не будут изменяться со временем.

Отобразим графики нашего ряда, чтобы наглядно в этом убедиться.

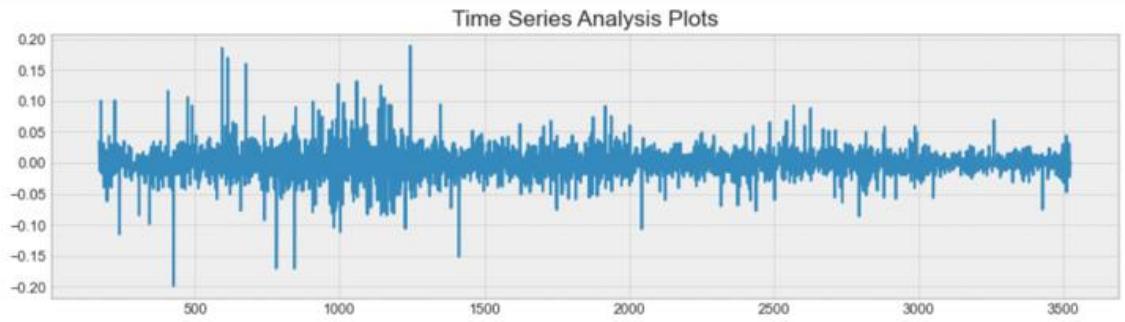

Рисунок 8 – График стационарного временного ряда

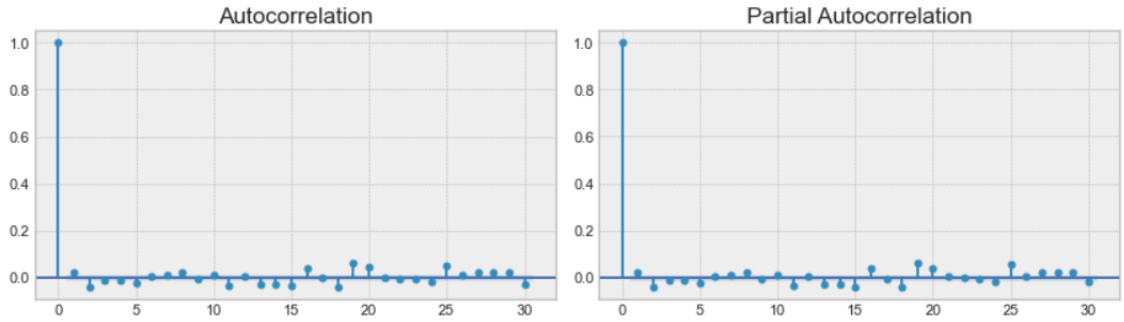

Рисунок 9 – Графики автокорреляции и частной автокорреляции

Автокорреляционная функция показывает зависимость автокорреляции от величины сдвига во времени. При этом предполагается стационарность временного ряда, означающая в том числе независимость автокорреляций от момента времени.

Можно начать моделировать процесс прогнозирования цены. Для начала необходимо обучить нашу модель SARIMA, в ходе анализа и обучения на исторических данный выявляются закономерности, которые в последствии используются для прогнозирования. Возьмем набор данных в период с 15.11.2004 по 14.11.2018.

В процессе работы модель обучается на тестовых данных, выявляя наилучшие параметры для составления прогноза.

```
Model:               SARIMAX(4, 1, 3)x(4, 1, 1, 24)
```

Рисунок 10 – Наилучшие параметры модели

3.3 Анализ полученных результатов

Итогом проделанной работы является график прогноза цены акций компании Amazon.

Модель оценила предыдущие значения, построив график поверх уже известных данных. На их основе было построено продолжение, позволяющее увидеть поведение акций на фондовом ранке в ближайший 51 месяц.

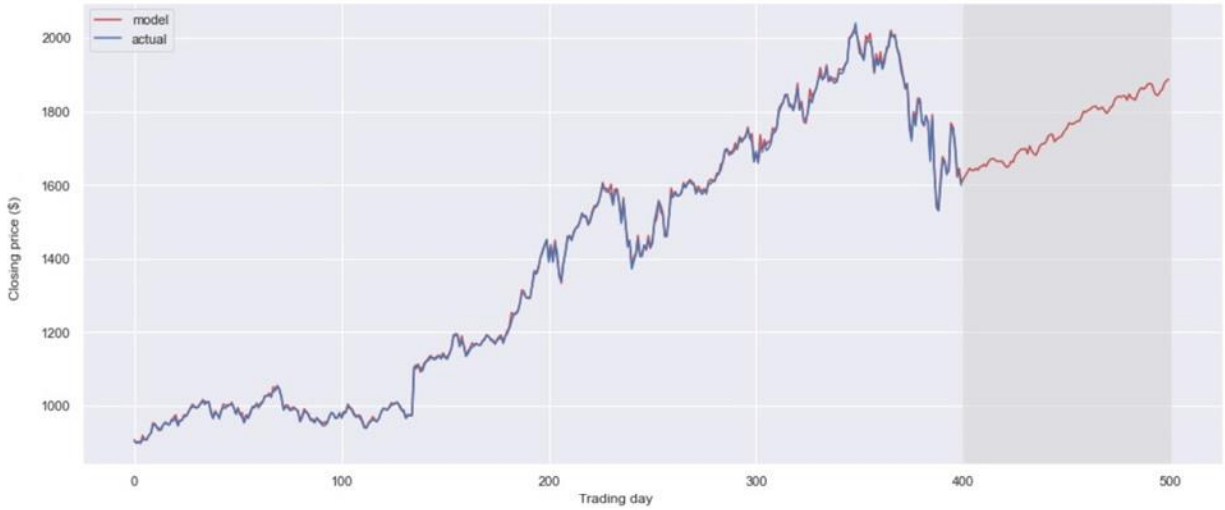

Рисунок 11 – Прогнозируемая цена

Синим цветом отображается график изменения цен на акции, взятых из исходного файла. Красным – данные обучения модели и дальнейший прогноз.

Глядя на него, можно сделать выводы о том, что в ближайшее время не предвидится серьезных спадов или резких скачков, а значит людям с долгосрочной инвестиционной стратегией стоит приобрести акции этой компании.

ЗАКЛЮЧЕНИЕ

В процессе работы изучен язык программирования Python3, библиотеки matplotlib, NumPy и Pandas, методы указанных библиотек для работы с временными рядами. Рассмотрен дистрибутив Anaconda и среда разработки Jupyter Notebook. Рассмотрены различные способы анализа, моделирования и прогнозирования временных рядов. Разработана программная реализация для сглаживания и оценивая временного ряда, построена модель обучения на данных с дальнейшим прогнозированием на конкретном примере данных компании Amazon. Полученные результаты хорошо согласуются с реальным поведением анализируемого временного ряда.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. 7 способов прогнозирования временных рядов с помощью питона. // [Электронный ресурс] – 2021. URL: http://distrland.blogspot.com/2019/09/7-python.html (дата обращения: 16.12.2021).
2. LSTM — нейронная сеть с долгой краткосрочной памятью. // [Электронный ресурс] – 2021. URL: https://neurohive.io/ru/osnovy-data-science/lstm-nejronnaja-set/ (дата обращения: 16.12.2021).
3. Анализ и тест Дикки-Фуллера // [Электронный ресурс] – 2021. URL: 7 способов прогнозирования временных рядов с помощью питона. // [Электронный ресурс] – 2021. URL: https://www.swe-notes.ru/post/time_series_analisys/ (дата обращения: 16.12.2021).
4. Временной ряд. // [Электронный ресурс] – 2021. URL: https://blog.skillfactory.ru/glossary/vremennoj-ryad-2/ (дата обращения: 11.11.2021).
5. Временные ряды с формулами. // [Электронный ресурс] – 2021. URL: http://rusforexclub.com/articles/24-finansovaya-matematika/99-vremennye-ryady#c6 (дата обращения: 11.11.2021).
6. Грас, Дж. DataScience. Наука о данных с нуля / Дж. Грас. – СПб.: БХВ-Петербург, 2017. – 336с. ISBN 978-5-9775-3758-2.
7. Коэльо, Л.П. Построение систем машинного обучения на языке Python / Луис Педро Коэлью, Вилли Ричарт. – М.: ДМК Пресс, 2016. - 302 с.: ил. ISBN 978-5-97060-330-7.
8. Модели прогнозирования временных рядов. // [Электронный ресурс] – 2021. URL: https://www.mbureau.ru/articles/dissertaciya-model-prognozirovaniya-vremennyh-ryadov-glava-1#p_1.3.1 (дата обращения: 16.12.2021).
9. Открытый курс машинного обучения. Тема 9. Анализ временных рядов с помощью Python // [Электронный ресурс] – 2021. URL: https://habr.com/ru/company/ods/blog/327242/ (дата обращения: 11.11.2021).

10. Прогнозирование акций+данные. // [Электронный ресурс] – 2021. URL: 7 способов прогнозирования временных рядов с помощью питона. // [Электронный ресурс] – 2021. URL: http://distrland.blogspot.com/2019/09/7-python.html (дата обращения: 16.12.2021).

11. Руководство: как использовать Python для алгоритмической торговли на бирже. Часть 1 // [Электронный ресурс] – 2021. URL: https://habr.com/ru/company/iticapital/blog/331542/ (дата обращения: 11.11.2021).

12. Халафян, А.А. STATISTICA 6. Статистический анализ данных. / А. А. Халафян А.А. – М.: ООО “Бином-Пресс”, 2007. – 512 с.: ил. ISBN 978-5-9518-0215-6.

13. Элбон, К. Машинное обучение с использованием Python. Сборник рецептов. / К. Элбон – СПб.: БХВ-Петербург, 2019. – 384 с.: ил. ISBN 978-5-9775-4056-8.

ПРИЛОЖЕНИЕ А

Программный код

```
import pandas as pd
import numpy as np
from pandas import read_csv, DataFrame
import matplotlib.pyplot as plt
%matplotlib inline
import scipy.integrate as integrate
import scipy
from scipy import stats
from scipy.stats import norm
import scipy.stats as stats
from datetime import datetime
import statsmodels.formula.api as smf
import statsmodels.api as sm

amazon=read_csv('AMZN.csv')
amazon
amazonClose=amazon.Close
amazonClose
plt.figure(figsize=(15, 7))
plt.plot(amazonClose)
plt.ylabel('Closing price ($)')
plt.xlabel('Trading day')
plt.grid(False)
test = sm.tsa.adfuller(amazonClose)
print ('adf: ', test[0])
print ('p-value: ', test[1])
```

```
if test[0]> test[1]:
    print ('есть единичные корни, ряд не стационарен')
else:
    print ('единичных корней нет, ряд стационарен')
#экспоненциальное сглаживание
def exponential_smoothing(series, alpha):
    result = [series[0]] # первое значение из серии
    for n in range(1, len(series)):
        result.append(alpha * series[n] + (1 - alpha) * result[n-1])
    return result
with plt.style.context('seaborn-white'):
    plt.figure(figsize=(20, 8))
    plt.plot(amazon.Close.values, label = "Actual")
    for alpha in [0.03,0.05]:
        plt.plot(exponential_smoothing(amazon.Close, alpha), label="Alpha
{}".format(alpha), color ='y')

    plt.legend(loc="best")
    plt.axis('tight')
    plt.title("Exponential Smoothing")
    plt.ylabel('Closing price ($)')
    plt.xlabel('Trading day')
    plt.grid(True)
test = sm.tsa.adfuller(amazonClose)
print ('adf: ', test[0])
print ('p-value: ', test[1])
if test[0]> test[1]:
    print ('есть единичные корни, ряд не стационарен')
else:
    print ('единичных корней нет, ряд стационарен')
```

```
#задание точки отсчета для разбиения данный на 2 части
m = amazonClose.index[int(len(amazonClose.index)/2+1)]
r1 = sm.stats.DescrStatsW(amazonClose[m:])
r2 = sm.stats.DescrStatsW(amazonClose[:m])
#ttest_ind()-встроенный метод, который возвращает p-значение
print ('p-value: ', sm.stats.CompareMeans(r1,r2).ttest_ind()[1])
def tsplot(y, lags=None, figsize=(12, 7), style='bmh'):
    if not isinstance(y, pd.Series):
        y = pd.Series(y)
    with plt.style.context(style):
        fig = plt.figure(figsize=figsize)
        layout = (2, 2)
        ts_ax = plt.subplot2grid(layout, (0, 0), colspan=2)
        acf_ax = plt.subplot2grid(layout, (1, 0))
        pacf_ax = plt.subplot2grid(layout, (1, 1))

        y.plot(ax=ts_ax)
        ts_ax.set_title('Time Series Analysis Plots')
        smt.graphics.plot_acf(y, lags=lags, ax=acf_ax, alpha=0.5)
        smt.graphics.plot_pacf(y, lags=lags, ax=pacf_ax, alpha=0.5)

        print("Критерий Дики-Фуллера: p=%f" % sm.tsa.stattools.adfuller(y)[1])

        plt.tight_layout()
    return

tsplot(amazonClose, lags=30)
amazonClose['season_diff'] = amazonClose_season - amazonClose_season.shift(1)
tsplot(amazonClose_season_diff[24*7+1:], lags=30)
ps = range(0, 5)
```

```
d=2
qs = range(0, 4)
Ps = range(0, 5)
D=1
Qs = range(0, 1)

from itertools import product

parameters = product(ps, qs, Ps, Qs)
parameters_list = list(parameters)
len(parameters_list)
parameters_list
from tqdm import tqdm
%%time
results = []
best_aic = float("inf")
for param in tqdm(parameters_list):
    #try except нужен, потому что на некоторых наборах параметров модель не обучается
    try:
        model=sm.tsa.statespace.SARIMAX(amazonnew.Close, order=(param[0], d, param[1]),
                                        seasonal_order=(param[2], D, param[3], 24*7)).fit(disp=-1)
    #выводим параметры, на которых модель не обучается и переходим к следующему набору
    except ValueError:
        print('wrong parameters:', param)
        continue
    aic = model.aic
```

```
    #сохраняем лучшую модель, aic, параметры
    if aic < best_aic:
        best_model = model
        best_aic = aic
        best_param = param
    results.append([param, model.aic])
warnings.filterwarnings('default')
result_table = pd.DataFrame(results)
result_table.columns = ['parameters', 'aic']
print(result_table.sort_values(by = 'aic', ascending=True).head())
%%time
best_model = sm.tsa.statespace.SARIMAX(amazonnew.Close, order=(4, d, 3), seasonal_order=(4, D, 1, 24)).fit(disp=-1)
print(best_model.summary())
tsplot(best_model.resid[24:], lags=30)
amazonClose["arima_model"] = invboxcox(best_model.fittedvalues, lmbda)
forecast = invboxcox(best_model.predict(start = amazonClose.shape[0], end = amazonClose.shape[0]+100), lmbda)
forecast = amazonClose.arima_model.append(forecast).values[-500:]
actual = amazonClose.values[-400:]
plt.figure(figsize=(15, 7))
plt.plot(forecast, color='r', label="model")
#plt.title("SARIMA model\n Mean absolute error {} users".format(round(mean_absolute_error(amazonClose.dropna().Close, amazonClose.dropna().arima_model))))
plt.plot(actual, label="actual")
plt.legend()
plt.axvspan(len(actual), len(forecast), alpha=0.5, color='lightgrey')
plt.grid(True)
```