МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ
имени И.Т. ТРУБИЛИНА»

Кафедра информационных систем и технологий

ТЕЗИС

ПОСЛЕДОВАТЕЛЬНОСТЬ ВЫПОЛНЕНИЯ ПРОГНОЗА И РАСЧЁТА ОШИБОК В ПРОГРАММНОЙ РЕАЛИЗАЦИИ ТРЕХЦВЕТНОЙ КЛЕТОЧНО-АВТОМАТНОЙ ПРОГНОЗНОЙ МОДЕЛИ

ПО КУРСОВОЙ РАБОТЕ

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ
ТРЁХЦВЕТНОЙ КЛЕТОЧНО-АВТОМАТНОЙ
ПРОГНОЗНОЙ МОДЕЛИ

ВЫПОЛНИЛ:
магистрант направления подготовки
09.04.03 Прикладная информатика
профиль «Менеджмент проектов
в области информационных систем»
Макаров Юрий Юрьевич

РУКОВОДИТЕЛЬ:
доктор экономических наук, кандидат физико-математических наук
Попова Елена Витальевна

Краснодар 2026

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В данном тезисе рассматривается практическая реализация алгоритма трехцветной клеточно-автоматной прогнозной модели в разработанном программном приложении.

Разработанное приложение реализует прогнозирование финансовых временных рядов (котировок акций) с использованием дискретной клеточно-автоматной модели. Архитектура программной системы построена по принципу разделения ответственности и включает:

- Data Layer - загрузка и сохранение данных
- Core Layer - алгоритм прогнозирования и расчёт метрик
- WpfApp Layer - пользовательский интерфейс и сценарий выполнения

Процесс прогнозирования организован как последовательный вычислительный конвейер.

1 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

1.1 Общая последовательность модели

1. Алгоритм функционирования реализован следующим образом:
2. Загрузка временного ряда цен
3. Расчёт логарифмических доходностей
4. Кодирование доходностей в трёхцветные состояния
5. Обучение вероятностных правил перехода состояний
6. Пошаговое прогнозирование будущих состояний
7. Восстановление прогнозных цен
8. Расчёт метрик точности прогноза
9. Сравнение моделей при различных глубинах памяти

1.2 Загрузка данных

Загрузка временного ряда осуществляется двумя способами:

- Импорт из CSV-файла
- Получение данных через API Московской биржи (MOEX ISS)

Результатом загрузки является структура данных, содержащая:

- массив цен закрытия x_t
- массив соответствующих дат

Дальнейшие вычисления производятся исключительно на основе массива цен.

Загрузка данных инициируется в файле CaForecast.WpfApp/MainViewModel.cs, где методы LoadCsv() и LoadMoexAsync() вызывают соответствующие сервисы получения данных. В файле CaForecast.Data/CsvImportService.cs метод Import() считывает CSV-файл, извлекает столбцы даты и цены закрытия и формирует объект CsvImportedData. В файле

CaForecast.Data/MoexIssService.cs метод ImportDailyHistory() выполняет HTTP-запрос к MOEX ISS, построчно обрабатывает полученный CSV-ответ и также формирует CsvImportedData. Объект CsvImportedData содержит массив цен закрытия и массив дат, при этом дальнейшие вычисления модели выполняются только на основе массива цен.

1.3 Расчет логарифмических доходностей

Для исключения трендовой составляющей и нормализации ряда используется логарифмическая доходность:

$$r_t = \ln(x_t / x_{t-1})$$

Расчёт логарифмических доходностей реализован в файле CaForecast.Core/ReturnCalculator.cs
в классе ReturnCalculator, метод CalculateLogReturns(List<double> prices).

В этом методе происходит проход по массиву цен закрытия prices, начиная со второго элемента, и для каждой пары соседних значений вычисляется логарифм отношения текущей цены к предыдущей через выражение:

Math.Log(current / previous);

Результатом работы метода является новый список логарифмических доходностей

1.4 Кодирование в трёхцветное состояние

Полученные доходности дискретизируются в три состояния - {-1, 0, 1}.

Состояние определяется следующим системой уравнений:

$s_t = 1$ при $r_tt > k$

$s_t = -1$ при $r_t < -k$

$s_t = 0$ при $|r_tt| \leq k$

s_t - состояние системы в момент времени t

k - порог дискретизации

Экономическая интерпретация:

1 - существенный рост;

-1 - существенное падение;

0 - незначительное изменение (боковое движение).

Кодирование доходностей в трёхцветное состояние реализовано в файле CaForecast.Core/ThreeColorEncoder.cs в классе ThreeColorEncoder, метод Encode(List<double> returns, double k).

В этом методе происходит последовательный проход по списку логарифмических доходностей returns. Для каждого значения выполняется проверка относительно порога k: если доходность больше k, в результирующий список состояний добавляется 1; если меньше -k — добавляется -1; во всех остальных случаях добавляется 0. Таким образом формируется дискретный ряд состояний {0,1,-1}, что полностью соответствующий заданной системе условий.

Полученный список состояний далее используется на этапе обучения модели переходов и формирования вероятностной структуры клеточного автомата.

1.5 Обучение модели переходов

Обучение реализовано в файле CaForecast.Core/CaRuleTrainer.cs, класс CaRuleTrainer, метод Train(List<int> states, List<double> returns, int memory, double alpha).

1.5.1 Формирование паттернов

Для заданной глубины памяти m формируется паттерн:

$$parrent_t = (s_{t-m}, t_{t-m+1}, \ldots, s_{t-1})$$

$parrent_t$ - последовательность предыдущих m состояний;

s_{t-1} - последнее известное состояние.

s_{t-m} - состояние системы m шагов назад

В методе Train() для каждого момента времени, начиная с индекса memory, вызывается вспомогательный метод BuildPattern(states, startIndex, memory). Он формирует строковый ключ, представляющий последовательность из m предыдущих состояний. Этот ключ используется как идентификатор шаблона в словаре переходов.

1.5.2 Подсчёт частот переходов

Для каждого паттерна подсчитывается частота перехода в состояние s_t.

Для каждого сформированного паттерна определяется текущее состояние s_t , которое преобразуется в индекс через метод ToIndex(). Далее в словаре Dictionary<string, double[]> увеличивается счётчик соответствующего перехода. Таким образом накапливаются частоты переходов паттерн - следующее состояние.

1.5.3 Лапласовское сглаживание

Для предотвращения нулевых вероятностей применяется сглаживание:

$$P(s \mid pattern) = (count_s + a) / N + 3a$$

$count_s$ - количество случаев, когда после данного шаблона наблюдалось сотояние s

a - параметр Лапласовского сглаживания

P(s | pattern) - условная вероятность перехода к состоянию s при заданном шаблоне

После подсчёта частот для каждого паттерна вызывается метод ApplyLaplace(counts, alpha), где к каждому счётчику прибавляется параметр сглаживания alpha, затем выполняется нормализация на сумму N + 3a. Это обеспечивает ненулевые вероятности даже для редких переходов и повышает устойчивость модели.

1.5.4 Глобальное распределение

Если паттерн отсутствует в обучающей выборке, используется глобальное распределение вероятностей: 1.0/3.0, 1.0/3.0, 1.0/3.0, что означает равную вероятность перехода. Необходимо для обеспечение устойчивости модели и предотвращение сбоев прогноза при встрече с ранее не наблюдавшейся комбинацией состояний.

1.5.5 Средние доходности по состояниям

Для каждого состояния рассчитывается:

$\hbar u_s = 1/n_s \ \sum r_t$

$\hbar u_s$ - средняя доходность для состояния s

N_s - количество наблюдений этого состояния.

Параллельно в методе Train() накапливаются суммарные частоты всех переходов в массиве globalCounts. После завершения обучения к ним также применяется лапласовское сглаживание, и формируется GlobalDistribution. Это распределение используется в прогнозе в случае, если встречается ранее неизвестный паттерн.

1.6 ПРОЦЕСС ПРОГНОЗИРОВАНИЯ

Прогноз осуществляется итерационно:

1. Формируется текущий паттерн.
2. Из модели извлекаются вероятности перехода.
3. Выбирается состояние с максимальной вероятностью.
4. Определяется прогнозная доходность:

$r_t = \hbar u_s * s$

s - состояние клеточного автомата (−1, 0 или +1), выбранное как наиболее вероятное.

$\hbar u_s$ - средняя доходность для состояния s

r_t - прогнозная доходность

5. Выполняется восстановление цены:

$\sim x_{t+1} = \sim x_t * e^{\sim r_t}$

$\sim x_t$ - текущая прогнозная цена;

$\sim x_{t+1}$ - следующая прогнозируемая цена;

r_t - доходность

Прогноз является рекурсивным — каждая новая цена зависит от предыдущей прогнозной.

В процессе обучения дополнительно рассчитываются средние логарифмические доходности для каждого состояния. Для этого в словаре MeanReturnsByState накапливаются суммы доходностей и количество наблюдений каждого состояния. Полученные значения используются на этапе прогноза для перевода предсказанного состояния в численную доходность.

1.7 РАСЧЁТ МЕТРИК ТОЧНОСТИ

Для оценки качества модели используются следующие показатели:

- средняя абсолютная ошибка(MAE) = $1/n\sum|x_t - \sim x_t|$
- среднеквадратичная ошибка(MSE) = $1/n\sum(x_t - \sim x_t)^2$
- корень из среднеквадратичной ошибки(RMSE) = sqrt(MSE)
- Средне абсолютная процентная ошибка(MAPE):

$100/n\sum|(x_t - \sim x_t)/x_t|)$

Метрики рассчитываются для цен, а не для доходностей.

Методы CalculateMae, CalculateMse, CalculateRmse и CalculateMape принимают два списка - фактические цены и прогнозные цены — и вычисляют соответствующие показатели

качества. В каждом методе выполняется проход по массивам значений, рассчитывается отклонение факта от прогноза и накапливается соответствующая агрегированная величина (абсолютная ошибка, квадрат ошибки или относительная ошибка). Метод CalculateRmse дополнительно берёт квадратный корень из значения MSE. Метрики рассчитываются именно для цен x_t и $\sim x_t$, а не для доходностей, поскольку конечной задачей модели является прогноз уровня цены.

1.8. ОЦЕНКА ГЛУБИНЫ ПАМЯТИ

Для различных значений m:

- выполняется обучение
- рассчитываются метрики
- выбирается модель с минимальной ошибкой.

Это позволяет определить оптимальную глубину памяти клеточного автомата для каждого временного ряда.

Оценка глубины памяти реализована в файле CaForecast.WpfApp/MainViewModel.cs, в методе RunAsync(bool showErrors, bool manageBusy = true).

В этом методе выполняется последовательный запуск прогноза для разных значений параметра memory. Для каждого значения глубины памяти вызывается обучение модели через CaRuleTrainer, затем выполняется прогноз через CaForecaster, после чего рассчитываются метрики точности. Результаты сохраняются, и выбирается вариант с наименьшим значением выбранной ошибки. Такой подход позволяет определить оптимальную глубину памяти клеточного автомата для конкретного временного ряда.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы была реализована трёхцветная клеточно-автоматная прогнозная модель для анализа финансовых временных рядов. Алгоритм включает последовательные этапы загрузки данных, расчёта логарифмических доходностей, дискретизации состояний, обучения вероятностной структуры переходов, пошагового прогнозирования и оценки точности полученных результатов.

Программная реализация выполнена с разделением на уровни Data, Core и WpfApp, что обеспечивает логическую структурированность и удобство сопровождения системы. Модель использует вероятностный механизм переходов состояний с лапласовским сглаживанием и механизм глобального распределения, что повышает её устойчивость при работе с ранее не наблюдавшимися паттернами.

Оценка качества прогноза осуществляется с использованием метрик MAE, MSE, RMSE и MAPE, что позволяет объективно сравнивать результаты при различных значениях глубины памяти. Проведённый анализ показывает, что выбор параметра памяти оказывает существенное влияние на точность прогнозирования и должен подбираться индивидуально для каждого временного ряда.

Разработанная система соответствует поставленным требованиям и позволяет формировать прогнозы для временных рядов произвольной длины с последующей количественной оценкой их точности.