

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Институт естественных и точных наук
Кафедра математического и компьютерного моделирования

РАБОТА ПРОВЕРЕНА Рецензент,
ген. директор ООО «Мегарендер»
_____/В.В. Юрков
« ____ » _____ 2019 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой,
д.ф.-м.н., доцент
_____/С.А. Загребина
« ____ » _____ 2019 г.

Анализ и прогнозирование динамики цен акций «Samsung Electronics Co.»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА ЮУрГУ –
02.03.01.2019.31.ПЗ ВКР

Руководитель работы, доцент
кафедры, к.ф.-м.н., доцент
_____/М.А Сагадеева
« ____ » _____ 2019 г.

Автор работы,
студент группы ЕТ-411
_____/А.С. Клепиков
« ____ » _____ 2019 г.

Нормоконтролер,
доцент кафедры, к.ф.-м.н.
_____/А.А. Акимова
« ____ » _____ 2019 г.

АННОТАЦИЯ

Клепиков А.С. – Анализ и прогнозирование цен акций «Samsung Electronics Co.» – Челябинск: ЮУрГУ, ЕТ-411, 38 с., 12 ил., библиогр. список – 20 наим., 1 прил.

Данная выпускная квалификационная работа посвящена разработке приложения, следящего за курсами цен акций «Samsung Electronics Co.», осуществляющего сбор данных об их изменениях и выполняющего анализ собранных данных для построения моделей, необходимых для прогнозирования цен в будущем.

Разработана математическая модель, построена архитектура приложения, реализованы аналитические алгоритмы. Разработан интерфейс для взаимодействия с программой и визуализации полученных данных.

Программа реализована на языке C# с использованием паттернов проектирования и принципов объектно-ориентированного программирования. Текст программы представлен в приложении.

ОГЛАВЛЕНИЕ

ВЕДЕНИЕ	5
1. ОБЩАЯ ИНФОРМАЦИЯ И МЕТОДЫ АНАЛИЗА ДАННЫХ	7
1.1. Информация о компании «Samsung Electronics Co.»	7
1.2. Информация об акциях	7
1.2.1. Категории акций	7
1.2.2. Стоимость акций	9
1.3. Общие сведения о биржах	10
1.4. Обзор методов прогнозирования данных	11
1.4.1. Простое скользящее среднее	11
1.4.2. Авторегрессионное скользящее среднее	12
1.5. Метод гусеницы	12
1.5.1. Развертка одномерного ряда в многомерный	13
1.5.2. Анализ главных компонент: сингулярное разложение выборочной ковариационной матрицы	13
1.5.3. Отбор главных компонент	14
1.5.4. Восстановление одномерного ряда	15
1.6. Выводы по разделу 1	16
2. РАЗРАБОТКА ПРИЛОЖЕНИЯ И ПРОГНОЗИРОВАНИЕ ЦЕН АКЦИЙ	17
2.1. Выбор средств разработки	17
2.1.1. Язык C#, платформа .Net Framework	17
2.1.2. Библиотеки	17
2.1.3. Система контроля версий Git	18
2.2. Написание алгоритма работы программы	20
2.2.1. Проектирование внутренней структуры, применение паттернов разработки	23
2.2.2. Разработка аналитических модулей	24
2.3. Разработка пользовательского интерфейса	24
2.4. Выводы по разделу 2	24
3. АНАЛИЗ И ПРОГНОЗИРОВАНИЕ ДАННЫХ	26

3.1. Реализация аналитических модулей	26
3.1.1. Блок-схема аналитических модулей	26
3.1.2. Пример работы программы	28
3.2. Анализ стоимостей акций «Samsung Electronics Co.»	31
3.3. Прогнозирование динамики изменения цен акций «Samsung Electronics Co.»	31
3.4. Выводы по разделу 3	34
ЗАКЛЮЧЕНИЕ	35
СПИСОК ЛИТЕРАТУРЫ	37
ПРИЛОЖЕНИЕ 1	39

ВВЕДЕНИЕ

Одним из популярных способов заработка является покупка-продажа акций на биржах. Это относительно простой способ на первый взгляд, но при ближайшем знакомстве становится понятно, что во многом всё зависит от статистики. Исследуя закономерности изменения прошлых изменений стоимости акций можно принимать решения о покупке-продаже в тот или иной момент. Но собирать данные вручную очень долгий и трудоёмкий процесс.

Для решения обозначенной выше задачи существуют уже готовые решения. Например, встроенный сервис технического анализа на сайте Investing.com [16]. Сервис прост в использовании, но имеет ограниченный набор инструментов для анализа данных. К тому же он имеет неявную реализацию, закрытую от пользователя. Также возможно собирать данные вручную и анализировать их в офисных программах, как например Microsoft Excel. Однако, в этом случае не имеется возможности автоматизировать сбор данных, но часть других функций потенциально автоматизируется макросами.

Целью данной работы является построение прогноза динамики изменения цен на акции компании «Samsung Electronics Co.». В качестве источника данных рассматривается Лондонская биржа, цены на которой представлены в USD.

Декомпозируем задачу. Для достижения вышеобозначенной цели необходимо решить следующие подзадачи:

- 1) провести обзор имеющихся методов прогнозирования;
- 2) выбрать наиболее подходящий метод;
- 3) собрать данные;
- 4) построить прогноз по имеющимся данным.

Задача о сборе данных и построение на их основе прогноза представляет собой ещё одну композицию. Для её решения будет разработано приложение, решающее следующие проблемы:

- 1) проектирование структуры приложения;
- 2) разработка жизненного цикла программы;
- 3) разработка алгоритма работы;

- 4) получение актуальной информации с сайта Investing.com в режиме реального времени;
- 5) запись и последующее чтение полученной информации в файл для оптимизации памяти и возможности выполнения своих функций даже после перезапуска;
- 6) реализация выбранных методов прогнозирования;
- 7) разработка пользовательского интерфейса программы;
- 8) вывод полученных и обработанных данных.

В первой главе будет рассмотрена общая информация об акциях, биржах и их различиях, а также рассмотрим подробнее существующие методы для прогнозирования данных, определив методы, которые будут использоваться в этой работе.

Вторая глава включит в себя разработку приложения, соответствующее обозначенным выше требованиям. В этом разделе будет рассмотрен процесс проектирования приложения, построения его алгоритма, разработки необходимых модулей и решения задачи пользовательского интерфейса. Также будут рассмотрены библиотеки, решающие те или иные задачи.

В третьей главе будут представлены результаты работы выбранных методов анализа и прогнозирования собранных данных.

1. ОБЩАЯ ИНФОРМАЦИЯ И МЕТОДЫ АНАЛИЗА ДАННЫХ

1.1. Информация о компании «Samsung Electronics Co.»

«Samsung Electronics Co.» является частью южнокорейской группы компаний Samsung Group, известной во всём мире как крупнейший производитель электроники. Ежегодно компания достигает отметки в более чем 400 миллионов проданных мобильных устройств, которые являются большей частью прибыли компании. Такие высокие продажи обусловлены высоким уровнем качества и технологий в выпускаемых продуктах.

Кроме производства уже готовых устройств, компания производит также комплектующие на продажу, такие как:

- 1) дисплеи, выполненные по технологии OLED и AMOLED, на которые «Samsung Electronics Co.» имеет эксклюзивные права;
- 2) NAND флеш-память;
- 3) литиево-ионные аккумуляторы и прочее;

Компания является крупным поставщиком для таких компаний как Sony, Apple, Dell, Hewlett-Packard и других крупных компаний.

Таким образом, «Samsung Electronics Co.» является одной из самых крупнейших компаний, не только производящей мобильные устройства, но и комплектующие для собственных нужд и на продажу прочим большим игрокам на рынке, что и обеспечивает высокую стоимость акций, год от года показывающих неуклонный рост.

1.2. Информация об акциях

1.2.1. Категории акций

Прежде чем рассматривать методы анализа данных и приступать к их реализации, рассмотрим, какие бывают акции и чем они отличаются [9]:

1. Обыкновенные акции дают право на участие в управленческих процессах общества (как правило, одна акция соответствует одному голосу на собрании акционеров, кроме проведения кумулятивного голосования) и участвуют в распределении полученной прибыли акционерного общества. Источником для выплаты дивидендов по таким акциям

является чистая прибыль общества. Размер самих дивидендов определяется советом директоров предприятия и рекомендуется собранию акционеров, которое способно только уменьшить или оставить неизменным размер дивидендов относительно рекомендованного советом директоров. Распределение дивидендов владельцев обыкновенных акций между собой осуществляется пропорционально вложенным средствам (в зависимости от количества приобретённых акций).

2. Привилегированные акции имеют право вносить ограничения на участие в управленческих процессах, а также могут давать дополнительные права в управлении, но по сравнению с обыкновенными акциями имеют ряд преимуществ: возможность получения гарантированного дохода, внеочерёдное выделение прибыли на выплату дивидендов, внеочерёдное погашение стоимости акции при ликвидации общества. Дивиденды, как правило, фиксированы в виде определённой доли от бухгалтерской чистой прибыли или в абсолютном денежном значении. Дивиденды по привилегированным акциям могут выплачиваться как из прибыли, так и из других источников — в соответствии с уставом акционерного общества. Привилегированные акции делятся на:

2.1. Привилегированные имеют ряд привилегий в обмен на право голоса. У их собственника определена величина дохода в момент выпуска и размещения ценных бумаг. Также у таких акций определён размер ликвидационной стоимости. Приоритет при переводе этих начислений по отношению к обыкновенным.

2.2. Кумулятивные (накапливающие). Привилегии — как у привилегированных. Сохраняется и обязательство по выплате дивидендов и оно же накапливается. Срок накопления дивидендов фиксирован. При невыплате дивидендов обладатели этого вида акций получают право голоса до выплаты дивидендов.

Аналог привилегированных акций — учредительская акция — акция, распространяемая среди учредителей акционерных компаний и дающая им некоторые преимущественные права. Держатели таких акций имеют возможность:

- 1) иметь дополнительное количество голосов во время собрания акционеров;
- 2) пользоваться внеочередным правом на получение акций в случае их последующих эмиссий;
- 3) играть главную роль в решении любых всех вопросов, которые связаны с деятельностью акционерных компаний.

По именованным акциям данные об их владельцах регистрируются в реестре акционерного общества. В соответствии с законодательством физические и юридические лица могут быть владельцами именных акций.

Акции на предъявителя допускают их свободную куплю-продажу на вторичном рынке без необходимости перерегистрации владельца. Российское законодательство допускало выпуск акций на предъявителя до 2002 года. С 2003 года акции могут выпускаться только в форме именной ценной бумаги.

1.2.2. Стоимость акций

Различают разные виды стоимости акций:

1. Номинальная стоимость акции — это то, что указано на её лицевой стороне (иногда её называют нарицательной стоимостью). Общая величина уставного капитала равна общей сумме номиналов всех выпущенных акций.

Номинальная стоимость всех обыкновенных акций общества должна быть одинаковой.

Номинальная стоимость не обязана отражать реальную ценность акций. Однако она часто используется для ряда операций (оценка пошлин, комиссий, тарифов), особенно на неразвитом, малоликвидном фондовом рынке. Цена акций при первичном размещении не должна быть ниже номинальной стоимости.

2. Эмиссионная стоимость акции — стоимость акций при их первичном размещении, по которой её приобретает первый держатель. Обычно эмиссионная цена акции превышает номинальную стоимость или равна ей. Превышение эмиссионной цены над номинальной стоимостью называется эмиссионной выручкой, или эмиссионным доходом.

3. Балансовая стоимость акций — частное от деления стоимости чистых активов компании (балансовой стоимости компании) на количество выпущенных акций, находящихся в обращении. Если рыночная цена ниже балансовой, то это является основой для будущего биржевого роста цены. Обычно балансовую стоимость определяют при аудиторских проверках.
4. Рыночная стоимость акции — это цена, по которой акция продаётся и покупается на вторичном рынке. Рыночная цена (котировка, курс) обычно формируется на торгах на фондовой бирже и отражает баланс спроса и предложения на данные акции. Для формирования рыночной цены важное значение имеет уровень ликвидности фондового рынка. Косвенно, рыночная стоимость акций отражает ликвидационную стоимость активов и пассивов компании.

1.3. Общие сведения о биржах

Фондовая биржа — это финансовый институт, обеспечивающий регулярное функционирование организованного рынка ценных бумаг.

В каждой стране мира с рыночной экономикой есть национальная фондовая биржа. Это организованный рынок для торговли ценными бумагами: акциями, облигациями и другими финансовыми инструментами.

Для допуска к торгам на бирже акции должны пройти процедуру листинга или быть допущены к торгам без прохождения процедуры листинга.

Участие акции в торгах позволяет эмитенту привлечь самый дешёвый и самый долгосрочный капитал, повысить стоимость компании, снизить стоимость заимствований, поднять свой престиж, осуществлять дополнительную рекламу через биржевые каналы и успешно размещать последующие выпуски.

В данной работе будут рассматриваться биржи NASDAQ (сокр. от англ. National Association of Securities Dealers Automated Quotation, читается как «Насдак» — Автоматизированные котировки Национальной ассоциации дилеров по ценным бумагам) — американская биржа, специализирующаяся на акциях высокотехнологичных компаний (производство электроники, программного обеспечения и т. п.) и корейская биржа (англ. Korea Exchange,

KRX) — крупнейшая в мире биржа по объему сделок с деривативами.

1.4. Обзор методов прогнозирования данных

Технический анализ – наиболее распространенный вид анализа валютного рынка. Он дарит ощущение полного контроля в прогнозах торгующего человека. Его суть заключается в анализе прошлого, таким образом трейдеры анализируют прошлые котировки и уровни цены, и на основе этого делают выводы о будущем [8].

Методология заключается в сборе данных и восстановлении ряда по определённым правилам. Таким образом, описав правило, по которому мы восстанавливаем значения ряда на основе предыдущих его значений [11], можно перейти к прогнозированию ряда и выявить общую закономерность образования новых данных [7].

Так как наше приложение свободно манипулирует большим количеством текущих и прошлых данных, этот вид анализа наиболее подходящий.

1.4.1. Простое скользящее среднее

Метод простого скользящего среднего (МА) основан на выборе некоторого числа последних элементов и подсчёте их среднего арифметического. Такое число считается следующим порядковым числом, идущим за выборкой. Модель скользящего среднего порядка q обозначается $MA(q)$ и записывается следующим образом [1]:

$$X_t = \sum_{i=1}^q \theta_i \varepsilon_{t-i} + \varepsilon_t,$$

где $\theta_1, \dots, \theta_q$ — параметры модели, а $\varepsilon_t, \dots, \varepsilon_{t-q}$ — ошибки.

Данный метод является очень простым в программной реализации и уже реализован во многих подключаемых пакетах. Несмотря на этот факт, метод даёт значения весьма близкие к действительным. Также метод простого скользящего среднего используется платформой Investing.com в качестве базового метода для прогнозирования будущих цен акций [6].

Тем не менее, несмотря на преимущества метода, он не позволяет построить достаточно точную модель, описывающую динамику поведения ряда в будущем, показывая себя лучше на точечных значениях, основанных на предыдущих значениях ряда [10].

1.4.2. Авторегрессионное скользящее среднее

Авторегрессионное скользящее среднее (ARMA), также называемая иногда моделью Бокса-Дженкинса, является объединением двух моделей исследования временных рядов: авторегрессионной (AR) и модели простого скользящего среднего (MA). Под обозначением ARMA(p,q) понимается модель, содержащая p авторегрессионных составляющих и q скользящих средних. Точнее модель ARMA(p,q) включает в себя модели AR(p) и MA(q) [14]:

$$X_t = c + \sum_{i=1}^q \theta_i \varepsilon_{t-i} + \sum_{i=1}^p \phi_i X_{t-i},$$

где ϕ_1, \dots, ϕ_p — параметры авторегрессионной модели, а c — константа. Для простоты константу, как правило, опускают [12].

Как правило, в качестве значения ошибки ε_t берут нормально случайные величины с нулевым средним: $\varepsilon_t \sim N(0, \sigma^2)$, где σ^2 — дисперсия [2].

Метод является улучшенным вариантом метода простого скользящего среднего и корректирует получаемые средние значения на основе авторегрессии ряда [3]. Метод позволяет получать более точные точечные значения, но также, как и предыдущий, не позволяет рассмотреть динамику ряда в достаточно большой перспективе с необходимой точностью [5].

1.5. Метод гусеницы

Метод гусеницы (SSA) в базовом варианте состоит в следующих шагах [4]:

- 1) развертка одномерного ряда в многомерный;
- 2) анализ главных компонент: сингулярное разложение выборочной ковариационной матрицы;

- 3) отбор главных компонент;
- 4) восстановление одномерного ряда.

Применив метод Гусеницы к исходному ряду, получим ряд, разложенный на простые компоненты. В результате это разложение может служить основой прогнозирования как самого ряда, так и его отдельных составляющих [13].

Рассмотрим временной ряд $x_{i=1}^N$, образованный последовательностью N значений некоторой (вероятно, случайной) функции $f(t)$:

$$x_i = f((i - 1)\delta t),$$

где $i = 1, 2, \dots, N$.

В действительности, существует несколько вариантов алгоритма, Опишем в этой работе один из них:

1.5.1. Развертка одномерного ряда в многомерный

Пусть есть некоторое число $M < N$, представляющее длину гусеницы, и положим первые M значений последовательности f в качестве первой строки матрицы X . Второй строкой матрицы будут являться значения последовательности с x_2 по x_{M+1} . Последней строкой с номером $k = N - M + 1$ будут последние M членов исходной последовательности. Получившуюся матрицу, элементы которой представляются как $x_{ij} = x_{i+j-1}$, можно рассматривать как M -мерную выборку объема k или M -мерный временной ряд, которому соответствует M -мерная траектория (ломаная в M -мерном пространстве из $k - 1$ звена.

Далее по обычной схеме (но без стандартизации признаков) выполняется анализ главных компонент (АГК).

1.5.2. Анализ главных компонент: сингулярное разложение выборочной ковариационной матрицы

Первым делом, вычисляется матрица $V = (1/k)X^T X$. Мы будем называть ее ковариационной матрицей, несмотря на то, что ее элементы не центрированы, иногда добавляя слово «нецентральная».

Далее, как обычно в АГК, вычисляются собственные числа и вектора матрицы V , т.е. разложение ее $V = P\Lambda P^T$, где Λ - это диагональная матрица, на диагонали которой стоят собственные числа, упорядоченные по убыванию, а P - ортогональная матрица собственных векторов матрицы V .

Матрицы L и P совместно имеют множество интерпретаций, основанных на АГК. Будем рассматривать матрицу P как матрицу перехода к главным компонентам $XP = Y = (y_1, y_2, \dots, y_M)$.

В случае, если изучается выборка из случайной совокупности, то собственные числа матрицы V являются выборочными дисперсиями соответствующих главных компонент, а квадратные корни из них - выборочными стандартами. Графическое представление собственных чисел и некоторых функций от них в АГК традиционно используется для выявления структуры исследуемой совокупности и отбора и интерпретации главных компонент.

Заметим также, что при выборе длины гусеницы, равной $N - M + 1$, собственные вектора и главные компоненты просто меняются местами с точностью до нормировки.

1.5.3. Отбор главных компонент

В силу свойств матрицы P возможно представить матрицу ряда X как $X = YP^T$. Таким образом, мы получаем разложение матрицы ряда по ортогональным составляющим, а именно - главным компонентам.

В то же время преобразование $y_j = Xp_j$ является линейным преобразованием исходного процесса с помощью дискретного оператора свертки, т.е.

$$y_j[l] = \sum_{q=1}^M x_{lq} p_{jq} = \sum_{q=1}^M x_{l+q-1} p_{jq}.$$

Таким образом, процедура «Гусеница» представляет собой набор линейных фильтров, настроенных на составляющие исходного ряда. При этом собственные векторы матрицы V выступают в качестве переходных функций соответствующих фильтров.

Аналитическое и визуальное изучение собственных векторов и главных компонент, полученных в результате линейной фильтрации, может дать

много интересной информации о структуре изучаемого процесса и свойствах составляющих его слагаемых.

В частности, среди главных компонент можно выделить:

1. относящиеся к тренду (медленно меняющиеся);
2. периодические;
3. шумовые.

Крайне эффективным выходит изучение построенных графиков изучение двумерных графиков, аналогичных фигурам Лиссажу, когда по осям x и y откладываются различные пары собственных векторов или главных компонент. Известно, что, если по осям откладывать значения синусоиды одной и той же частоты, но с разными фазами, то на плоскости получается эллипс. Из того факта, что собственные вектора и главные компоненты ортогональны следует, что фазовый сдвиг между такими парами наверняка будет равен $\pm\pi/2$ и, таким образом, эллипс переходит в окружность.

1.5.4. Восстановление одномерного ряда

Заключительным и наиболее важным этапом метода «Гусеница» является процедура восстановления. Этот этап основан на разложении $X = YP^T$. Восстановление проводится по данному набору главным компонентам, если при применении формулы восстановления $X = YP^T$ матрица Y получена из матрицы Y обнулением всех не входящих в набор главных компонент. Таким образом, становится возможным получить интересующее нас приближение матрицы ряда или интерпретируемую часть этой матрицы.

Переход к восстановленному ряду возможно осуществить усреднением матрицы ряда по побочным диагоналям, но в этом случае возможно некоторое искажение полученной структуры.

Отметим сразу одну очень важную в этой работе отличительную особенность описанного метода - его интерактивность. То есть возможность реализовать общение пользователя и программы. Интерактивность метода связана с типично статистическим свойством алгоритма, а именно, необходимостью интерпретации промежуточных результатов и управлением выполнением алгоритма в процессе многоэтапной процедуры обработки. Однако, несмотря на это преимущество, оно нам не потребуется. Анализ по-

лученных данных будет производиться посредством изучения визуальной модели.

Из-за высокой точности динамики получаемых моделей воспользуемся методом Гусеницы в качестве основного метода для анализа данных воспользуемся именно им.

1.6. Выводы по разделу 1

В первом разделе содержится подробное исследование предмета последующего изучения в этой работе - акций, бирж и самой компании «Samsung Electronics Co.».

Были рассмотрены виды акций и образование их стоимости. Также были рассмотрены биржи и, в частности, рассмотрена корейская биржа, на которой и будет базироваться исследование, проводимое в этой выпускной квалификационной работе.

В главе также были изучены методы прогнозирования и анализа временных рядов. Было решено преимущественно использовать метод гусеницы для анализа временного ряда по причине высокой точности к прогнозированию динамики временного ряда.

2. РАЗРАБОТКА ПРИЛОЖЕНИЯ И АНАЛИЗ И ПРОГНОЗИРОВАНИЕ ЦЕН АКЦИЙ

2.1. Выбор средств разработки

Для разработки аналитического приложения, способного самостоятельно собирать данные с интернет-ресурса в режиме реального времени нужна развитая платформа с большим набором инструментов. Для этих целей как раз подходит платформа Microsoft .Net Framework. Эта платформа реализует большой инструментарий для работы как с анализируемыми данными, так и с более абстрагированными объектами.

2.1.1. Язык C#, платформа .Net Framework

На сегодняшний день язык программирования C# и платформа .Net Framework являются одними из наиболее популярных инструментов у разработчиков по всему миру. На этой платформе можно реализовать совершенно разные решения: от простейших консольных программ до сложнейших научных библиотек и микросервисов в веб-стеке [15].

Язык проектировался как полностью объектно-ориентированный, и его компания-разработчик Microsoft всегда выделяет это свойство языка в качестве его главных преимуществ. Ещё одним преимуществом является сравнительно низкий порог входа, обеспечивая вместе с тем большую потенциальную мощность разрабатываемых приложений.

Ещё одним преимуществом языка C# и платформы .Net Framework является развитое сообщество. Этот факт выводит возможности встроенного пакетного менеджера NuGet на новый уровень: любой желающий может разработать своё собственное решение задачи и загрузить его на сервер Microsoft в открытый доступ. В данной работе воспользуемся этой возможностью и загрузим библиотеки как необходимые для дальнейшей работы, так и упрощающие реализацию некоторых решений.

2.1.2. Библиотеки

В разработанном приложении использованы следующие библиотеки:

1. HtmlAgilityPack. Данный пакет позволяет получать html-код веб-страницы и предлагает удобные методы для парсинга DOM-дерева с помощью XPath-строки [18];
2. MathNet. Пакет, в котором реализованы объекты для работы с линейной алгеброй, статистикой, теорией вероятностей и некоторые другие а также методы для работы с ними. Также реализована перегрузка операторов для новых типов данных. Воспользуемся данным пакетом для работы с матрицами в методе Гусеницы [19];
3. Стандартные библиотеки .Net Framework. Среди таких библиотек есть библиотеки для [20]:
 - 3.1. сериализации (чтения и записи) собранных данных в .json-файл для дальнейшей работы с ними;
 - 3.2. работы со стандартными коллекциями, такими как список;
 - 3.3. получения необходимой выборки из стандартных коллекций и их сортировки;
 - 3.4. работы с WinForms.

2.1.3. Система контроля версий Git

Система контроля версий — это система, записывающая изменения в файл или набор файлов в течение времени и позволяющая вернуться позже к определённой версии.

Ядро Git представляет собой набор команд для командной строки с параметром. Все настройки хранятся в текстовых файлах. Такая система позволяет сделать Git легко портируемым на любую платформу и легко интегрировать в любую систему.

Система имеет низкий порог входа для пользователей, предлагая управление файлами как с помощью командной строки, так и с помощью различных пользовательских интерфейсов. Справедливо будет сказать, что низкий порог входа обеспечивается преимущественно для людей, достаточно глубоко знакомых с информационными технологиями, в то время как остальным нужно будет потратить больше времени и усилий, чтобы понять, как работать с Git.

Основной концепцией для понимания работы с Git является разделения рабочего пространства на 3 условных зоны (2.1).

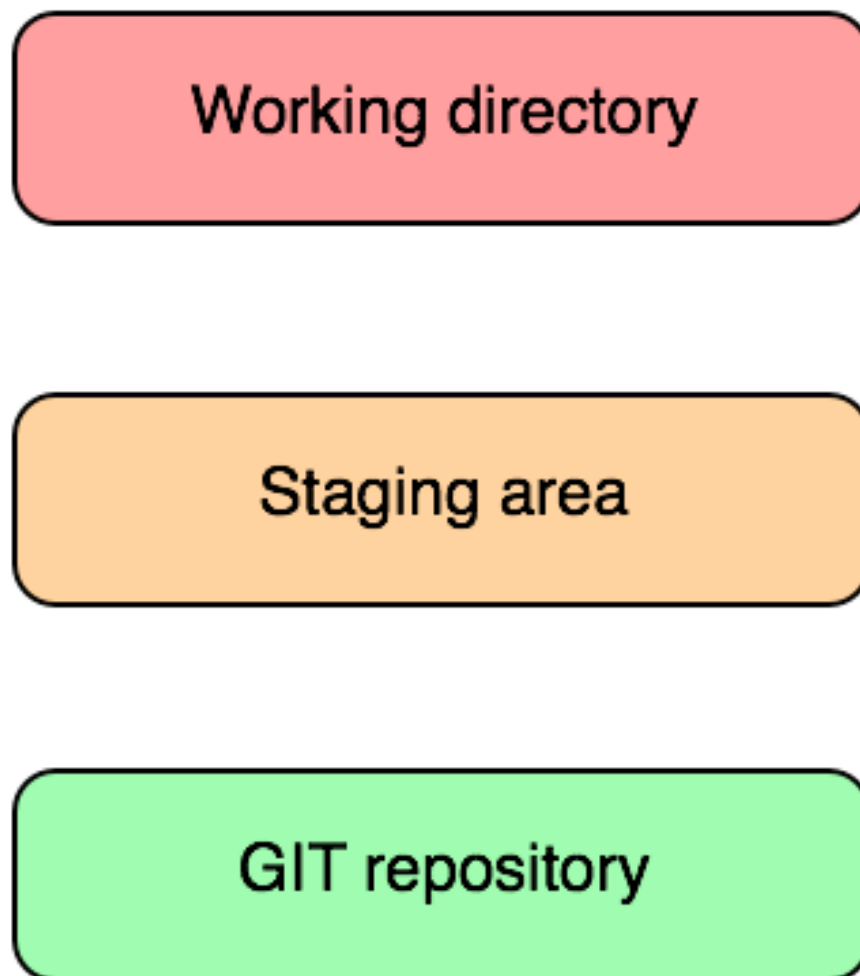


Рисунок 2.1 – Рабочее пространство Git

В качестве первой зоны выступает рабочий каталог (Working directory). Он представляет собой буквально каталог, в котором мы работаем и вносим изменения в существующие файлы, удаляем их или создаём новые.

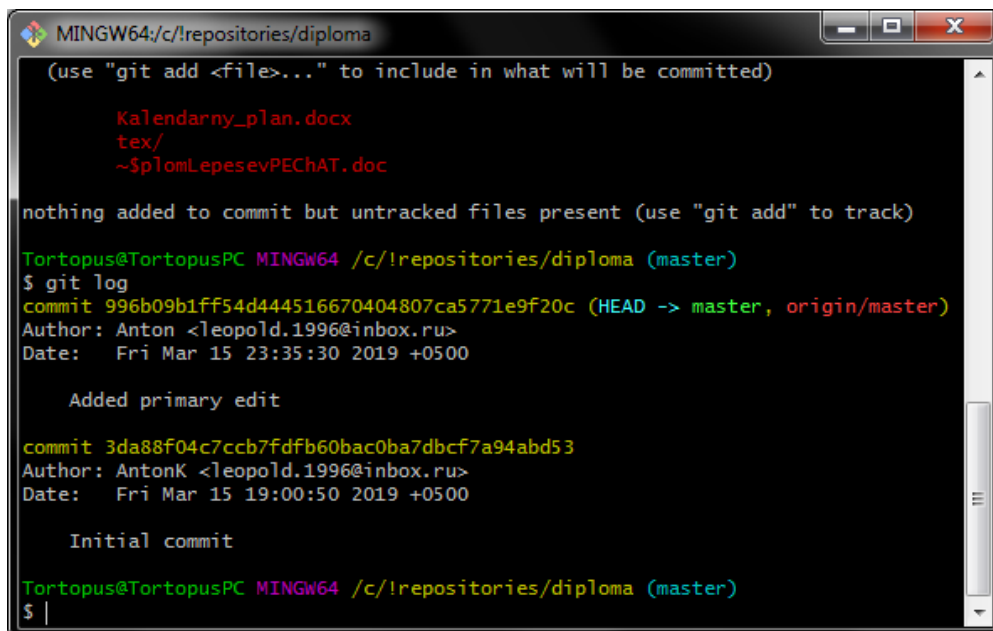
Внеся необходимые изменения, мы выбираем интересующие нас файлы. Таким образом, файлы попадают в промежуточную зону (Staging area).

Когда выбор файлов окончен, изменения фиксируются, и оказываются в репозитории (Git repository).

Репозиторий Git представляет собой каталог на диске, содержащий файлы проекта и файлы конфигурации. Сам репозиторий может быть на удалённом сервере после вносимых изменений для возможности работы на

разных устройствах и для повышения безопасности во избежания непредвиденных обстоятельств, связанных с аппаратной частью компьютера.

Управление изменениями и репозиторием происходит посредством текстовых команд из командного окна: «git add», «git commit», «git push», «git pull» и так далее. Пример представлен на рисунке 2.2.



```
MINGW64:/c:/repositories/diploma
(use "git add <file>..." to include in what will be committed)

Kalendarny_plan.docx
tex/
~$plomLepesevPEChAT.doc

nothing added to commit but untracked files present (use "git add" to track)

Tortopus@TortopusPC MINGW64 /c:/repositories/diploma (master)
$ git log
commit 996b09b1ff54d444516670404807ca5771e9f20c (HEAD -> master, origin/master)
Author: Anton <leopold.1996@inbox.ru>
Date:   Fri Mar 15 23:35:30 2019 +0500

    Added primary edit

commit 3da88f04c7ccb7fd6b60ba7dbcf7a94abd53
Author: AntonK <leopold.1996@inbox.ru>
Date:   Fri Mar 15 19:00:50 2019 +0500

    Initial commit

Tortopus@TortopusPC MINGW64 /c:/repositories/diploma (master)
$ |
```

Рисунок 2.2 – Командное окно Git Bash

Также Git поддерживает ветвление. То есть, возможно создание ветки от определённой фиксации (коммита) для внесения тематических изменений. При таком подходе в определённый момент образуется множество параллельных веток, несущих разные изменения. Все эти изменения возможно объединить посредством слияния (merge). Слияние пройдёт в автоматическом режиме в том случае, если в файлах нет конфликтов изменений. В случае возникновения таких конфликтов необходимо будет вручную их разрешить, оставив желаемые изменения, после чего зафиксировать результат слияния.

2.2. Написание алгоритма работы программы

Для разработки приложения, которое будет самостоятельно собирать данные с удалённых источников и в последствии их анализировать, необходимо реализовать следующие возможности:

1. сбор данных непосредственно с удалённой веб-страницы;
2. сериализация собранных данных;
3. асинхронное чтение собранных ранее данных и запись их в память;
4. прогноз будущих значений;
5. отображение полученных данных на пользовательской форме.

Приложение разработано под ОС Windows, для запуска необходимо выполнить исполняемый файл.

При запуске программы инициализируется пользовательская форма. При нажатии на кнопку «Начать» начинается цикл сбора данных.

Программа каждую минуту параллельно начинает попытки подключения к сайту Investing.com. Получив страницу, приложение использует XPath строку для выделения нужной информации о цене акций и состоянии площадки.

После получения цены на акции происходит запись в .json-файл всего списка значений, а также добавление полученного значения в сам список.

Далее, когда данные собраны, при построении графика, данные читаются из файла, и на их основе строится два графика: с исходным рядом и прогнозируемым.

Блок-схема рабочего цикла представлена на рисунке 2.3.

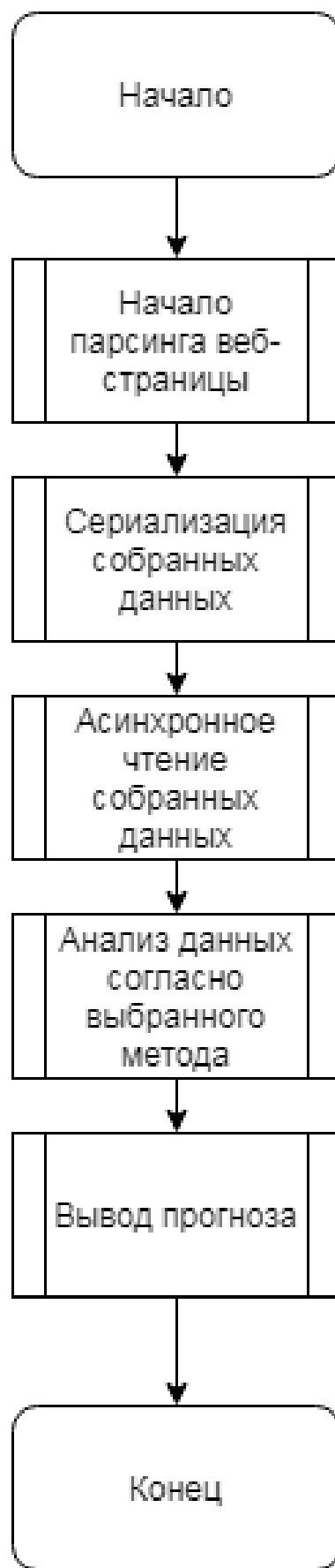


Рисунок 2.3 – Основной алгоритм программы

2.2.1. Проектирование внутренней структуры, применение паттернов разработки

Основной рабочей структурой в программе являются аналитические модули. Они организованы в соответствии с паттерном проектирования «фабричный метод». На языке диаграмм UML можно представить структуру следующим образом (2.4).

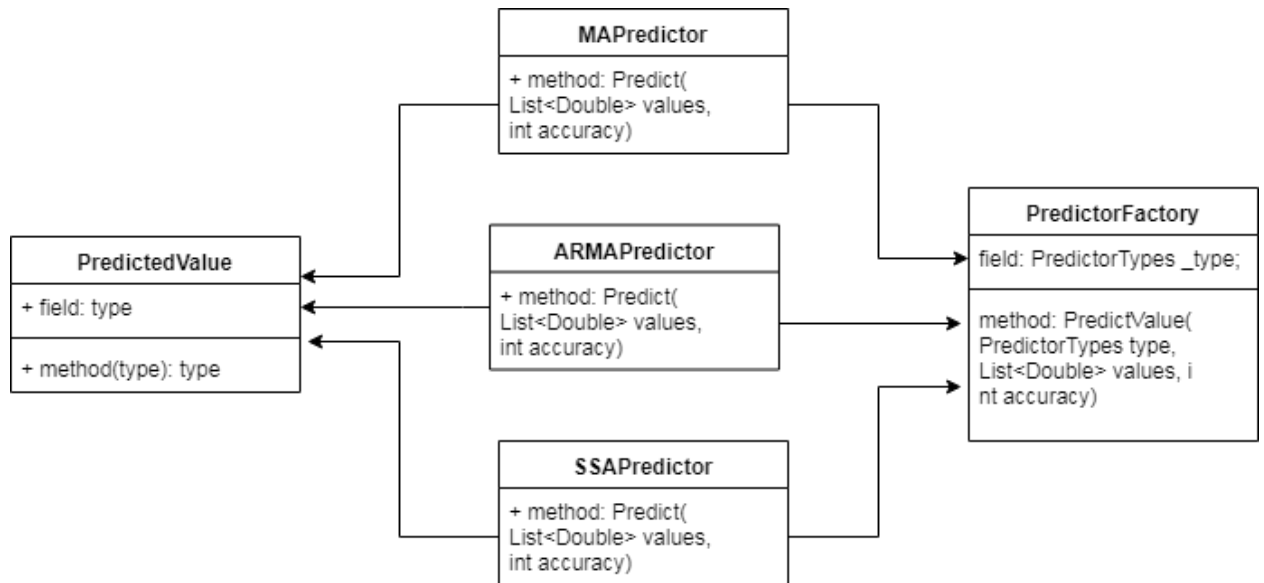


Рисунок 2.4 – Описание паттерна проектирования «фабричный метод»

2.2.2. Разработка аналитических модулей

В программе имеется три аналитических модуля, спроектированных по паттерну разработки «фабричный метод» реализующие различные прогнозирующие методы:

1. метод простого скользящего среднего;
2. метод авторегрессионного скользящего среднего;
3. метод гусеницы.

Доступ к модулям осуществляется из класса-сборщика PredictorFactory. На вход помещается временной ряд, требуемая точность прогнозирования и тип прогнозирующего метода.

Далее, в зависимости от типа, выбирается соответствующий метод, и ему в обработку передаётся временной ряд и заданная точность.

2.3. Разработка пользовательского интерфейса

Для базового управления процессом и вывода результатов пользователю используется технология WinForms.

Для построения пользовательской формы были использованы стандартные компоненты, предлагаемые .Net Framework:

1. кнопки;
2. текстовое поле;
3. график;
4. выпадающий список;
5. текстовая подпись.

Элементы были скомпонированы на форме для максимально удобного пользования приложением и для удобства просмотра информации.

2.4. Выводы по разделу 2

Из обзора видно, что необходима платформа, которая сможет совместить в себе возможность получения информации из интернет-ресурсов и мощные аналитические инструменты для работы с собранными данными.

В качестве платформы для разработки был выбран язык программирования C# на платформе .Net Framework, так как он отвечает всем необходимым требованиям.

Также важным этапом разработки является версионный контроль для сохранения данных и восстановления их в случае утери. Сегодня версионный контроль представляет неотъемлемую часть разработки и используется опытными разработчиками на любых проектах, вне зависимости от их масштаба.

Основным методом для анализа данных был выбран метод Гусеницы ввиду высокой точности прогнозируемой динамики на большом масштабе, в чём ему методы простого и авторегрессионного скользящего среднего проигрывают.

В результате было разработано приложение, следящее за изменением цен на акции «Samsung Electronics Co.», и анализирующее собранные результаты тремя разным различными методами.

3. АНАЛИЗ И ПРОГНОЗИРОВАНИЕ ДАННЫХ

3.1. Реализация аналитических модулей

Для реализации описанных выше методов был создан ряд классов, отражающих сущности операций и математических объектов. Классы организованы по шаблону проектирования «фабричный метод». В этом случае вызов нужного метода располагается в реализующем его классе и проходит через транзитный класс, определяющий используемый метод анализа на основе входящего параметра, определяющего тип метода для анализа.

3.1.1. Блок-схема аналитических модулей

На блок-схеме (рисунок 3.1) можно увидеть процесс поступления на обработку данных, процесса выбора нужного метода прогнозирования и возврата прогнозируемого значения.

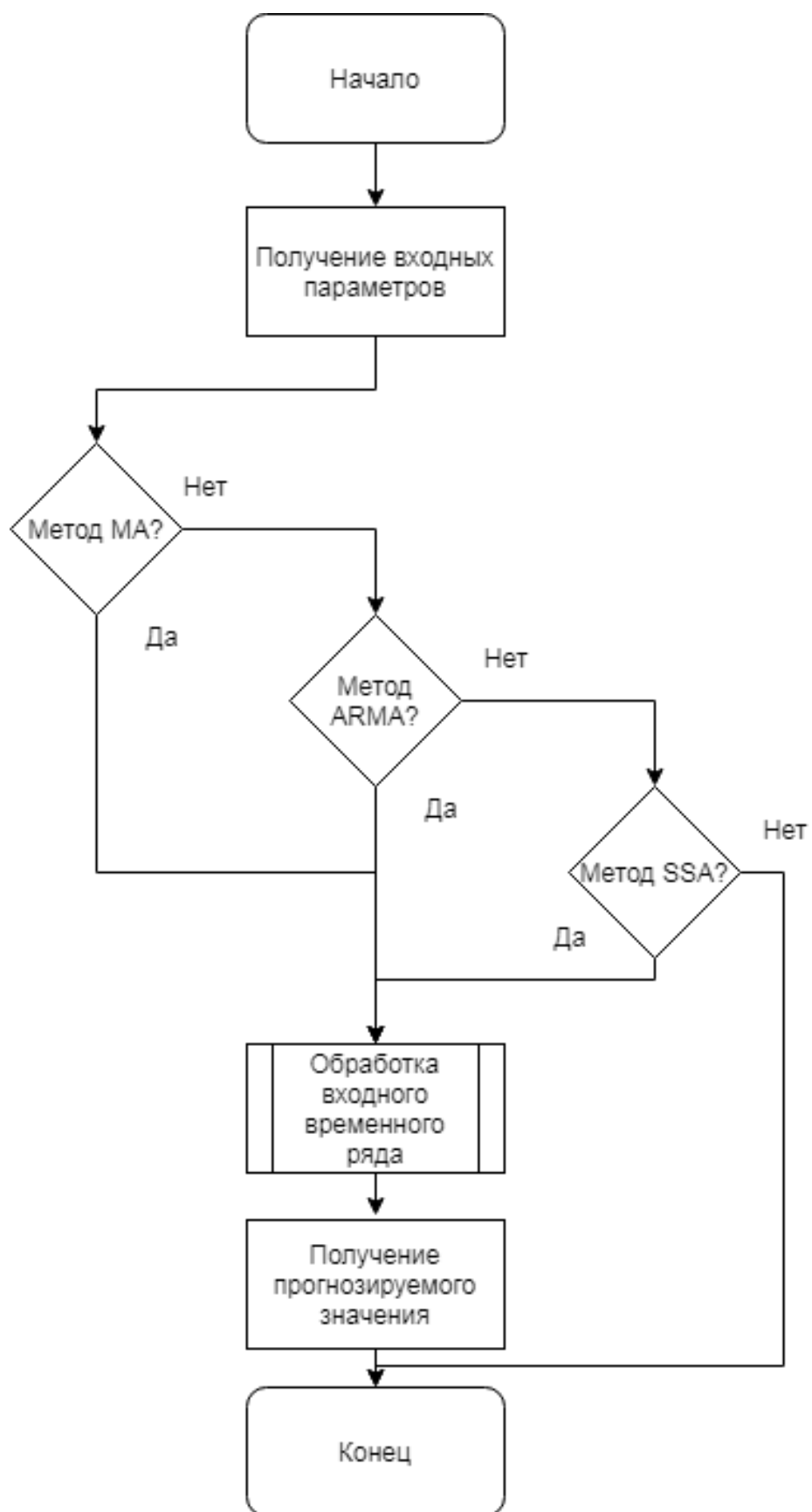


Рисунок 3.1 – Блок-схема работы аналитических модулей

3.1.2. Пример работы программы

При запуске программа инициализирует форму, заполняет внутренний список значений из прилагающегося .json-файла с данными, собранными ранее, если таковые имеются. Инициализировавшись, форма приобретает следующий вид(рис. 3.2):

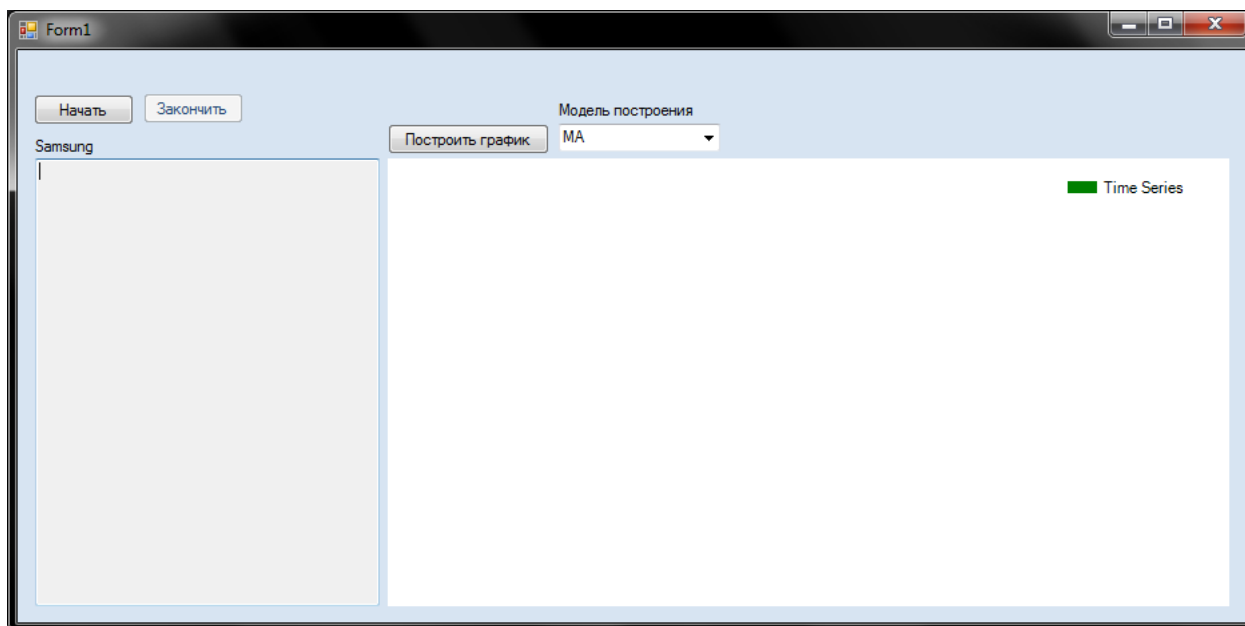


Рисунок 3.2 – Главная форма программы

В правой части программы располагается место под график на основе уже собранных данных. Выпадающий список предлагает выбрать метод прогнозирования, по умолчанию это простое скользящее среднее. После выбора графика необходимо нажать на прилагающую кнопку, чтобы построить график. Пример на изображён на рисунке 3.3.

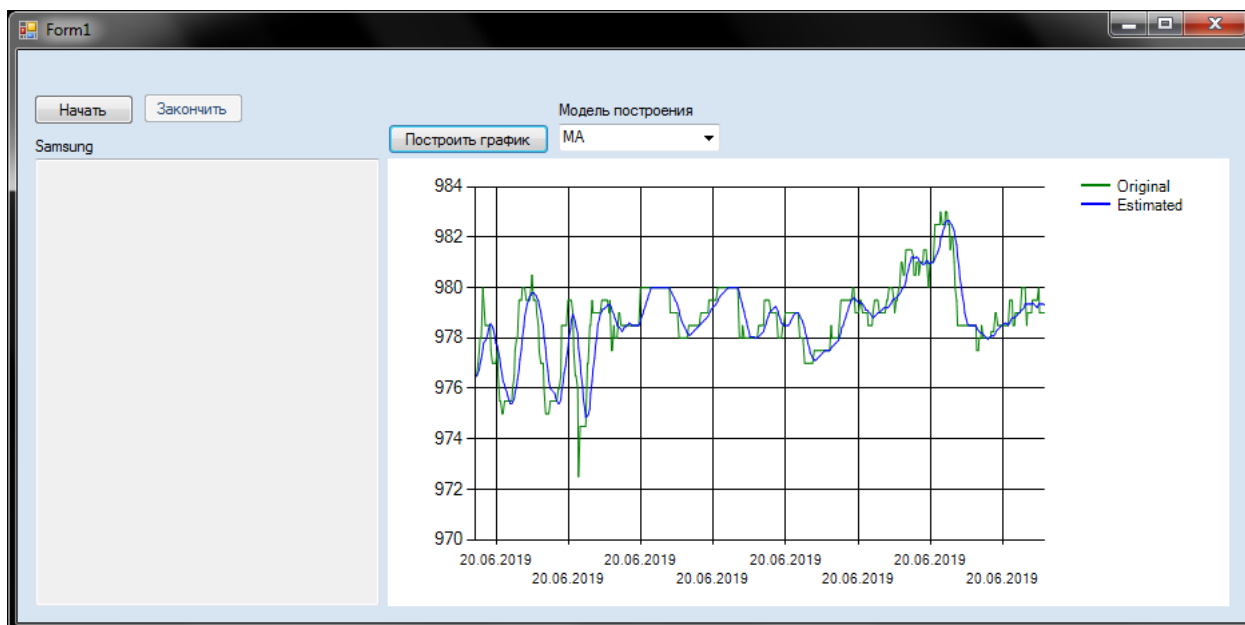


Рисунок 3.3 – Пример построения графика

Для того, чтобы построить новый график, необходимо в выпадающем списке выбрать интересующую модель (рисунок 3.4) и снова нажать на кнопку «Построить график».

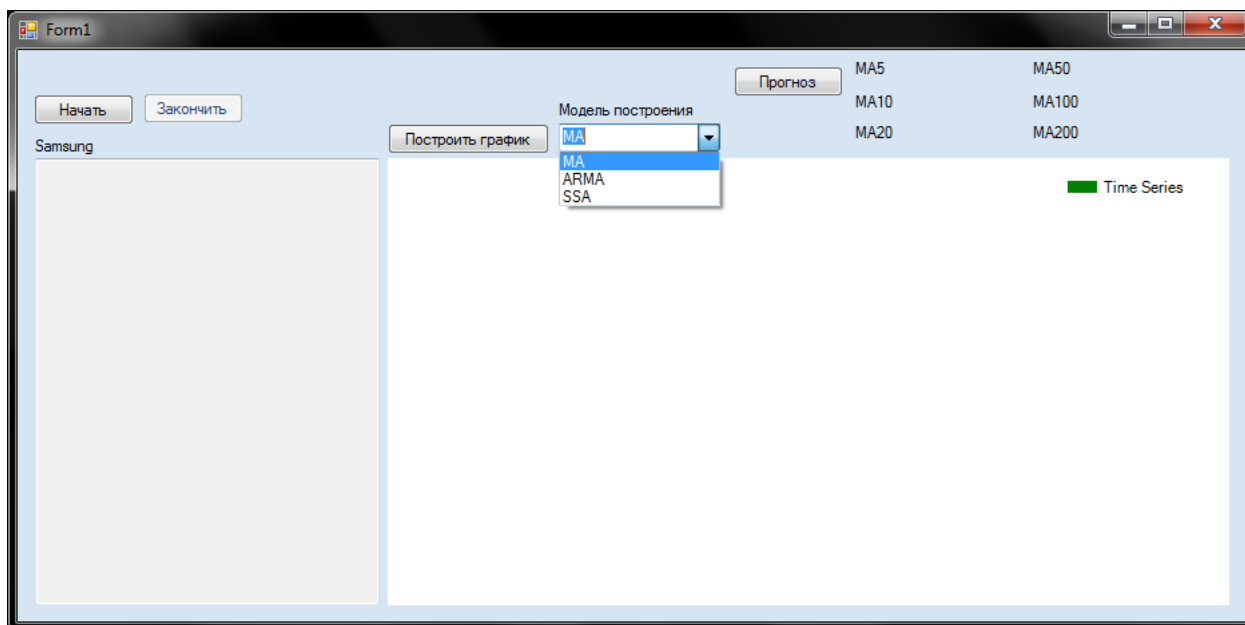


Рисунок 3.4 – Выпадающий список, предлагающий выбрать модель

Выберем для примера модель ARMA и нажмём кнопку «Построить график». Результат этих действий на рисунке 3.5.

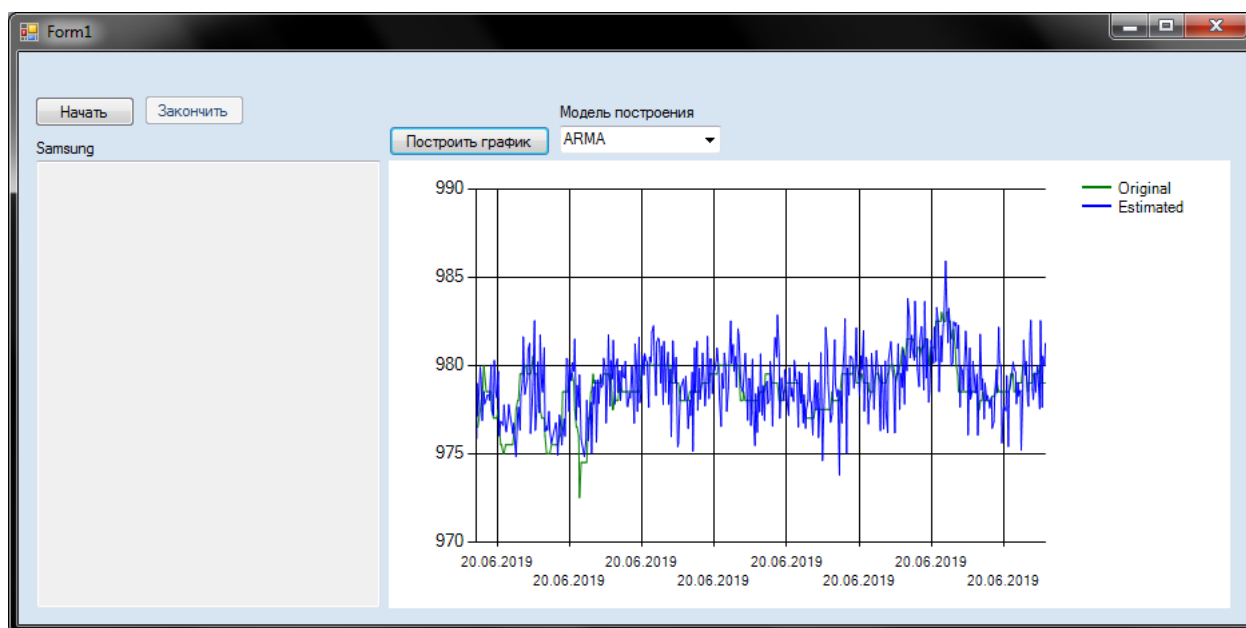


Рисунок 3.5 – Выпадающий список, предлагающий выбрать модель

3.2. Анализ стоимостей акций «Samsung Electronics Co.»

Анализ стоимостей акций показал, что наиболее эффективным методом оказался метод «Гусеницы», который выполняет построение максимально точно к исходному ряду. Пример такого построения изображён на рисунке 3.6.

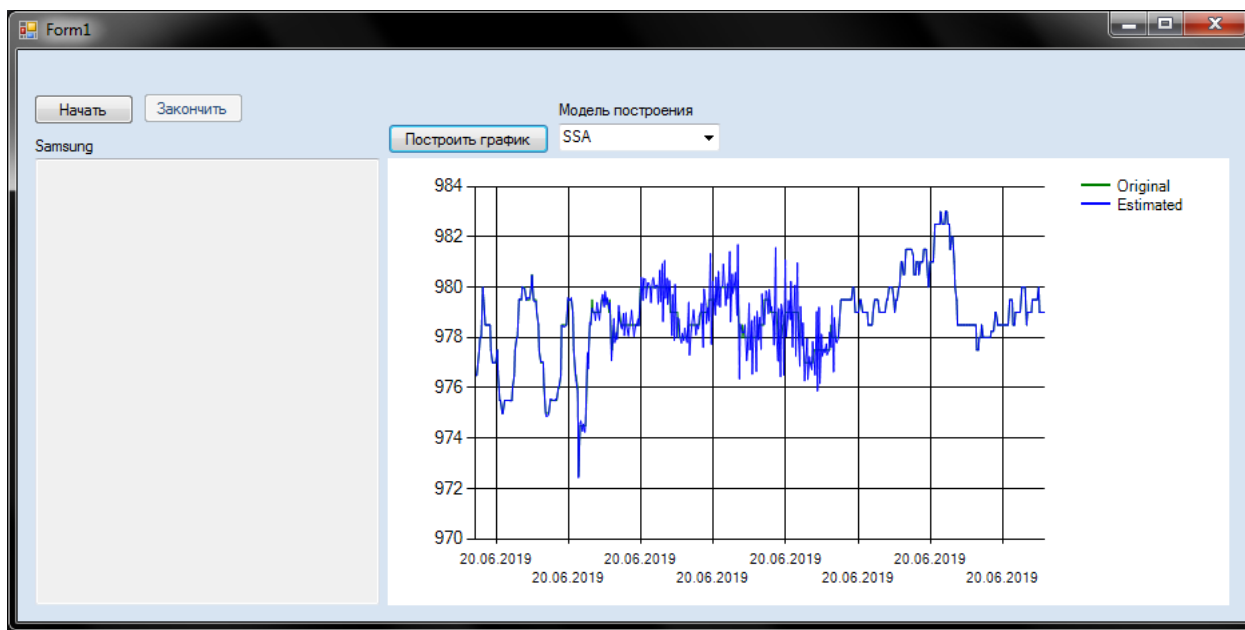


Рисунок 3.6 – График, основанный на модели «Гусеница»

На графике видно, что построенная модель практически полностью перекрывает исходный ряд, что говорит о её высоком качестве. Такая близость значений объясняется преимущественно линейным поведением временного ряда в исследуемый промежуток. Не исключено, что при более резких колебаниях значений расхождения были бы более существенны.

3.3. Прогнозирование динамики изменения цен акций «Samsung Electronics Co.»

Для прогнозирования будущих значений ряда можно воспользоваться любым из описанных в предыдущей главе методом. Возьмём данные, собранные на торгах за 20 июня 2019 года. В качестве метода прогнозирования воспользуемся методом простого скользящего среднего, потому как этот метод наиболее прост в программной реализации и цель состоит в кратко-

срочном прогнозе динамики. Воспользуемся собранными значениями для прогнозирования данных (рисунок 3.7):

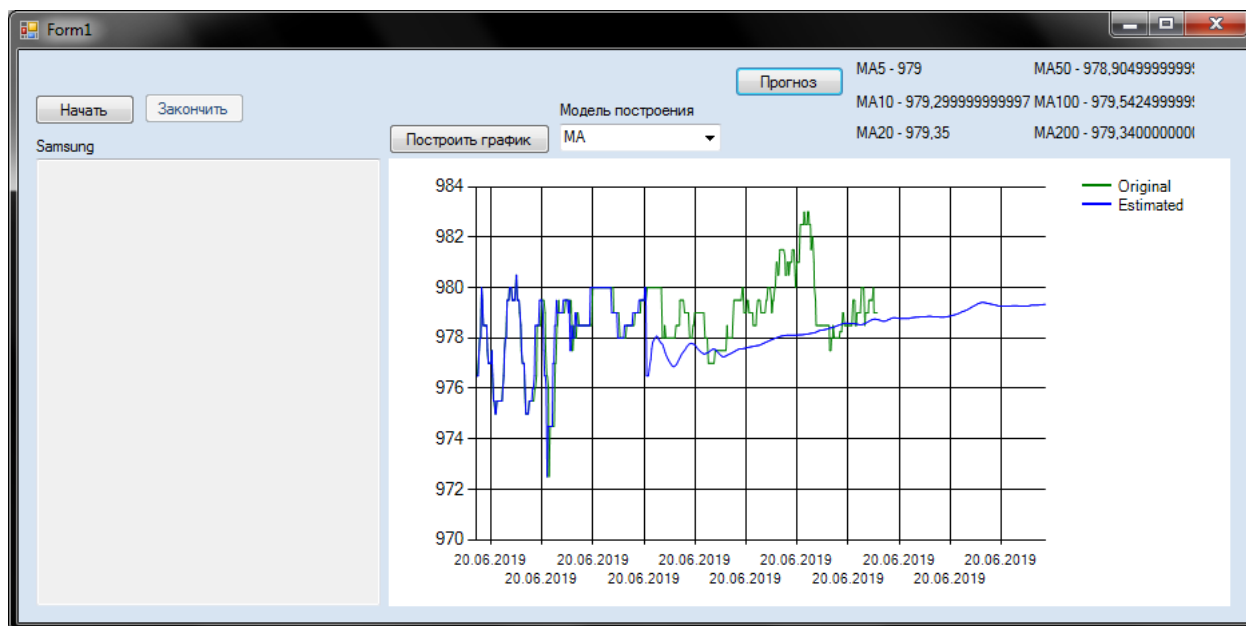


Рисунок 3.7 – График спрогнозированной динамики изменения цен на акции «Samsung Electronics Co.»

Сравним результаты с техническим анализом на сайте Investing.com (рисунок 3.8).

Скол. средние » 20.06.2019 15:40 GMT

Период	Простая	Экспоненциальная
MA5	979 Покупать	978,32 Покупать
MA10	979,29 Покупать	976,76 Покупать
MA20	979,35 Покупать	975,38 Покупать
MA50	978,91 Покупать	967,50 Покупать
MA100	979,55 Покупать	1.049,79 Продавать
MA200	979,34 Продавать	1.126,72 Продавать
Покупать: 9 Продавать: 3 Резюме: ПОКУПАТЬ		

Рисунок 3.8 – Технический анализ от сайта Investing.com

Как видно, данные полностью совпадают, что говорит о том, что сервис Investing.com действительно использует методы простого скользящего среднего для прогноза будущих значений.

3.4. Выводы по разделу 3

В этом разделе было рассмотрено разработанное для целей данной работы приложение. Показаны примеры работы приложения на собранных данных с сайта Investing.com.

В результате анализа данных был построен прогноз с помощью метода простого скользящего среднего. Сравнение построенного прогноза с имеющимся на сайте Investing.com показало, что наш прогноз полностью соответствует таковому на сайте.

ЗАКЛЮЧЕНИЕ

Данная выпускная квалификационная работа посвящена всегда актуальной теме исследования временных рядов на примере цен акций на основе собранных данных.

Целью работы являлось построение прогноза динамики изменения цен на акции «Samsung Electronics Co.».

В соответствии с целью были рассмотрены варианты методов прогнозирования и реализовано приложение, способное в автономном режиме собирать актуальные данные и анализировать их. В результате сбора и анализа данных был построен собственный прогноз, полностью совпадающий с таковым на сайте Investing.com.

В первой главе была рассмотрена общая информация об изучаемой компании, акциях, формировании на них цен и биржах. Также в главе были рассмотрены методы прогнозирования и анализа данных. Для анализа было решено использовать метод Гусеницы по причине высокой точности к прогнозированию динамики.

Во втором разделе был построен алгоритм работы программы и разработана математическая модель взаимодействия программных компонентов между собой. Также было построено приложение для сбора данных, их последующего анализа, прогнозирования будущих значений на основе уже собранных и визуализации полученных данных.

В третьей главе было подробно рассмотрено приложение и его возможности, и с помощью реализации был проведён анализ собранных данных о ценах акций «Samsung Electronics Co.». Также был построен прогноз на основе модели простого скользящего среднего в разном диапазоне. Выявлено, что на сайте Investing.com используются те же инструменты для прогнозирования, что и описанные в этой работе.

В результате разработки было получено простое в использовании приложение, позволяющее:

1. автономно собирать информацию о состоянии площадки;
2. получать и записывать информацию об изменениях цен на акции;
3. проводить различные типы анализа данных;

4. выводить на график результаты анализа.

В ходе работы были решены следующие задачи:

1. выполнен обзор существующих методов прогнозирования;
2. разработан алгоритм работы программы;
3. разработан способ сериализации данных для последующего чтения;
4. реализован пользовательский интерфейс;
5. реализовано отображение данных.

Таким образом, все поставленные цели были достигнуты. В дальнейшем планируется добавление и реализация новых методов прогнозирования в программу и расширение функционала.

СПИСОК ЛИТЕРАТУРЫ

1. Айвазян С.А. Прикладная статистика. Основы эконометрики. Том 2. - М.: Юнити-Дана, 2001. — 432 с
2. Бардасов С. А., ЭКОНОМЕТРИКА: учебное пособие. 2-е изд., перераб. и доп. Тюмень: Издательство Тюменского государственного университета, 2010. 264 с.. 2010.
3. Доугерти К. Введение в эконометрику: Пер. с англ. - М.: ИНФРА-М, 1999. - 402 с.
4. Жиглявский А. А., Солнцев В. Н.: «Главные компоненты временных рядов: метод Гусеница».
5. Кремер Н. Ш., Путко Б. А. Эконометрика. - М.: Юнити-Дана, 2003—2004. - 311 с.
6. Магнус Я.Р., Катышев П.К., Пересецкий А.А. Эконометрика. Начальный курс. - М.: Дело, 2007. - 504 с.
7. Мишулина О. А. Статистический анализ и обработка временных рядов. - М.: МИФИ, 2004. - С. 180.
8. Роберт Колби. Энциклопедия технических индикаторов рынка - М.: Альпина Паблишер, 2018. - 837 с.
9. Тихомиров Н. П., Дорохина Е. Ю.; Рос. экон. акад. им. Г. В. Плеханова. - 2-е изд., стер. - М.: Экзамен, 2007. - 510 с.
10. Тихомиров Н. П., Эконометрика учеб. для вузов по специальности «Мат. методы в экономике»
11. Шмойлова Р. А. Общая теория статистики: Учебник. - М.: Финансы и статистика, 2002.
12. Эконометрика. Учебник / Под ред. Елисеевой И.И. - 2-е изд. - М.: Финансы и статистика, 2006. - 576 с.

13. Главные компоненты временных рядов: метод Гусеница, доступ: <http://www.gistatgroup.com/gus/book1/algor.html>.
14. Официальная документация к языку программирования C#, доступ: <http://www.machinelearning.ru/wiki/index.php?title=ARMA> (дата обращения – 25 июня).
15. Руководство по языку C#, доступ: <https://metanit.com/sharp/> (дата обращения – 25 июня).
16. Investing.com, доступ: <https://investing.com/> (дата обращения – 21 июня 2019).
17. Git - Book, доступ: <https://git-scm.com/book/en/v2>.
18. GitHub.com, доступ: <https://github.com/zzzprojects/html-agility-pack/tree/master/docs2/pages/documentations> (дата обращения – 7 июня 2019).
19. MathDotNet.com, доступ: <https://www.mathdotnet.com/> (дата обращения – 25 июня).
20. Microsoft.com, доступ: <https://docs.microsoft.com/ru-ru/dotnet/csharp/> (дата обращения – 25 июня).

ПРИЛОЖЕНИЕ 1

Form1.cs

```
using HtmlAgilityPack;
using System;
using System.Collections.Generic;
using System.IO;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting;
using HtmlDocument = HtmlAgilityPack.HtmlDocument;
using System.Runtime.Serialization.Json;
using System.Threading;
using MetalParser.Predicting;
using System.Linq;
using MathNet.Numerics.Statistics;

namespace MetalParser
{
    public partial class Form1 : Form
    {
        System.Windows.Forms.Timer tSamsung = new System.Windows.Forms
        int timeout = 60000;

        string platinum_path = "@/data/platinum-cfd.txt";
        string gold_path = "@/data/gold-cfd.txt";
        string silver_path = "@/data/silver-cfd.txt";
        string samsung_path = "@/data/Samsung-electronincs-Co.txt";
        string samsungJsonData = "@/data/samsung.json";
        string apple_path = "@/data/Apple.txt";

        List<double> platinumValues = new List<double>();
        List<double> goldValues = new List<double>();
```

```

List<double> silverValues = new List<double>();
List<Data> samsungValues = new List<Data>();
List<double> appleValues = new List<double>();

public Form1()
{
    InitializeComponent();
}

private async Task<string> FindValue(string Url, Object sender)
{
    string parsedValue = null;
    CancellationTokensource cancelTokensource = new Cancellati
    CancellationToken token = cancelTokensource.Token;
    await Task.Run(async () =>
    {
        try
        {
            HtmlWeb web = new HtmlWeb();
            HtmlDocument doc = web.Load(Url);
            if (doc != null)
            {
                string openState = doc.DocumentNode.SelectSingleNode
                parsedValue = openState.ToLower().Contains("3a
                if (parsedValue == "closed")
                {
                    cancelTokensource.Cancel();
                }
            }
        }
        catch (Exception ex)
        {
            parsedValue = "lost connection";
        }
    }
}

```



```

        }
    });
    if(cancelTokenSource.IsCancellationRequested)
    {
        tSamsung.Interval = timeout * 10; //Уменьшение частоты
        return null;
    }
    return parsedValue;
}

/// <summary>
/// Метод асинхронно получает значение акции и записывает его
/// </summary>
/// <param name="url">Ссылка на страницу со значением стоимости
private async void GetValue(string url, Object sender, EventArgs e)
{
    string value = await FindValue(url, sender, e);
    DateTime date = DateTime.Now;
    if (value == null)
    {
        textBox1.Text += $"{date.ToString("dd.MM.yy hh:mm")} | "
        return;
    }

    value = value.Replace(".", "");
    string line = $"{date.ToString("dd.MM.yy hh:mm")} | {value}

    if (value != "lost connection")
    {
        Data data = new Data(date, Double.Parse(value));
        samsungValues.Add(data);
        WriteValue(samsungJsonData, data);
    }
}

```

```

        textBox1.Text += line + Environment.NewLine;
    }

    /// <summary>
    /// Пишет полученное значение в json-файл
    /// </summary>
    /// <param name="path">Путь к json-файлу</param>
    /// <param name="data">Дата и цена акции</param>
    private void WriteValue(string path, Data data)
    {
        List<Data> values = ReadValues(path);
        values.Add(data);

        using (FileStream fs = new FileStream(samsungJsonData, FileMode.Open))
        {
            DataContractJsonSerializer jsonFormatter = new DataContractJsonSerializer(typeof(Data));
            jsonFormatter.WriteObject(fs, values);
        }
    }

    /// <summary>
    /// Читает json-файл со значениями даты и цены и возвращает список
    /// </summary>
    /// <param name="path">Путь к json-файлу</param>
    /// <returns></returns>
    private List<Data> ReadValues(string path)
    {
        List<Data> valuesList = new List<Data>();
        DataContractJsonSerializer jsonFormatter = new DataContractJsonSerializer(typeof(Data));
        using (FileStream fs = new FileStream(path, FileMode.Open))
        {
            List<Data> dataList = jsonFormatter.ReadObject(fs) as List<Data>;
        }
    }

```

```

        foreach (Data data in dataList)
        {
            valuesList.Add(data);
        }
    }

    return valuesList;
}

private void button1_Click(object sender, EventArgs e)
{
    button1.Enabled = false;
    button2.Enabled = true;

    tSamsung.Interval = timeout;
    tSamsung.Tick += (timer, arguments) => GetValue(samsungUrl);
    tSamsung.Start();
}

private void button2_Click(object sender, EventArgs e)
{
    button1.Enabled = true;
    button2.Enabled = false;
    tSamsung.Stop();
}

private void Form1_Load(object sender, EventArgs e)
{
    try
    {
        comboBox1.SelectedIndex = 0;
        //При загрузке формы читаем файл и заполняем список зн

```

```

        foreach(Data data in ReadValues(samsungJsonData))
        {
            samsungValues.Add(data);
        }
    }
    catch(Exception ex)
    {

    }
}

private void label1_Click(object sender, EventArgs e)
{

}

private void textBox1_TextChanged(object sender, EventArgs e)
{

}

private void button3_Click(object sender, EventArgs e)
{
    bar.ChartAreas.Clear();
    bar.ChartAreas.Add(new ChartArea("Time series"));
    Series series = new Series("Original");
    series.ChartType = SeriesChartType.Line;
    for (int i = 0; i < samsungValues.Count; i++)
    {
        series.Points.AddXY(samsungValues[i].Date, samsungValu

    }

    int option = comboBox1.SelectedIndex;

```

```

PredictorFactory predictorFactory = null;
switch (option)
{
    case 0:
        predictorFactory = new PredictorFactory(PredictorT
        break;

    case 1:
        predictorFactory = new PredictorFactory(PredictorT
        break;

    case 2:
        predictorFactory = new PredictorFactory(PredictorT
        break;
}

```

```

Series estimation = new Series("Estimated");
estimation.ChartType = SeriesChartType.Line;
List<Double?> predictedValues = new List<Double?>();
int accuracy = 10;
if (predictorFactory.Type == PredictorTypes.MA)
{
    List<Double> predictedValues1 = new List<double>();
    for (int i = 0; i < samsungValues.Count; i++)
    {
        predictedValues1.Add(samsungValues[i].Value);
    }
    List<Double> copy = MAPredictor.PredictList(predictedV

    for (int i = 0; i < samsungValues.Count; i++)
    {
        estimation.Points.AddXY(samsungValues[i].Date, copy
    }
}

```

```

    }
    else if (predictorFactory.Type == PredictorTypes.ARMA)
    {
        List<Double> copy = new List<Double>();
        for (int i = 0; i < samsungValues.Count; i++)
        {
            copy.Add(samsungValues[i].Value);
        }

        List<Double> prediction = ARMAPredictor.PredictList(copy);

        for (int i = 0; i < samsungValues.Count; i++)
        {
            estimation.Points.AddXY(samsungValues[i].Date, prediction[i]);
        }
    }
    else if (predictorFactory.Type == PredictorTypes.SSA)
    {
        List<Double> temp = new List<Double>();
        for (int i = 0; i < samsungValues.Count; i++)
        {
            temp.Add(samsungValues[i].Value);
        }

        List<double> predictedValues1 = SSAPredictor.PredictList(temp);

        for (int i = 0; i < samsungValues.Count; i++)
        {
            estimation.Points.AddXY(samsungValues[i].Date, predictedValues1[i]);
        }
    }

    bar.ChartAreas[0].AxisY.Minimum = 970;

    bar.Series.Clear();

```

```

        bar.Series.Add(series);
        bar.Series.Add(estimation);
    }

private void ComboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    panel1.Visible = comboBox1.SelectedIndex == 0;
    button4.Visible = comboBox1.SelectedIndex == 0;
}

private void Button4_Click(object sender, EventArgs e)
{
    bar.ChartAreas.Clear();
    bar.ChartAreas.Add(new ChartArea("Time series"));
    Series series = new Series("Original");
    series.ChartType = SeriesChartType.Line;
    Series estimation = new Series("Estimated");
    estimation.ChartType = SeriesChartType.Line;

    List<Double> values = new List<Double>();
    for (int i = 0; i < samsungValues.Count; i++)
    {
        series.Points.AddXY(samsungValues[i].Date, samsungValues[i].Value);
        values.Add(samsungValues[i].Value);
    }

    List<Double> estimated = values.MovingAverage(200).ToList();
    DateTime date = samsungValues[0].Date;
    for (int i = 0; i < samsungValues.Count + 200; i++)
    {
        if (i < 200)
            estimation.Points.AddXY(date, values[i]);
        else
            date = date.AddDays(1);
    }
}

```

```

        estimation.Points.AddXY(date, estimated[i-200]);

        date = date.AddMinutes(1);
    }

    MA5.Text = $"MA5 - {values.MovingAverage(5).ToList<Double>}";
    MA10.Text = $"MA10 - {values.MovingAverage(10).ToList<Double>}";
    MA20.Text = $"MA20 - {values.MovingAverage(20).ToList<Double>}";
    MA50.Text = $"MA50 - {values.MovingAverage(50).ToList<Double>}";
    MA100.Text = $"MA100 - {values.MovingAverage(100).ToList<Double>}";
    MA200.Text = $"MA200 - {values.MovingAverage(200).ToList<Double>}";

    bar.ChartAreas[0].AxisY.Minimum = 970;

    bar.Series.Clear();
    bar.Series.Add(series);
    bar.Series.Add(estimation);
}
}
}

```

PredictorFactory.cs

```

using System;
using System.Collections.Generic;

namespace MetalParser.Predicting
{
    class PredictorFactory
    {
        public PredictorTypes Type;

        public PredictorFactory(PredictorTypes type)
        {

```



```

        Type = type;
    }

    public double? PredictValue(List<Double> values, int accuracy)
    {
        double? predictedValue = null;

        switch (Type)
        {
            case (PredictorTypes.MA):
                predictedValue = MAPredictor.Predict(values, accuracy);
                break;

            case (PredictorTypes.ARMA):
                predictedValue = ARMAPredictor.Predict(values, accuracy);
                break;

            case (PredictorTypes.SSA):
                predictedValue = SSAPredictor.Predict(values, accuracy);
                break;
        }

        return predictedValue;
    }
}

```

PredictorTypes.cs

```

namespace MetalParser.Predicting
{
    enum PredictorTypes
    {
        MA,
    }
}

```

```

        ARMA,
        SSA
    }
}

```

MAPredictor.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using MathNet.Numerics.Statistics;

namespace MetalParser.Predicting
{
    class MAPredictor
    {
        /// <summary>
        /// Predicts next number using MA methond with defined accuracy
        /// </summary>
        /// <param name="values">Time series</param>
        /// <param name="accuracy">Number of last items in time series</param>
        /// <returns></returns>
        public static double Predict(List<Double> values, int accuracy)
        {
            Double maSum = 0;
            int window = values.Count / accuracy;

            for (int i = values.Count - 1; i > window; i--)
            {
                maSum += values[i];
            }
            return maSum/accuracy;
        }
    }
}

```

```

        public static List<Double> PredictList(List<Double> values, int accuracy)
        {
            var a = values.MovingAverage(accuracy);
            return a.ToList<Double>();
        }
    }
}

```

ARMAPredictor.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using MathNet.Numerics.Distributions;
using MathNet.Numerics.Statistics;

namespace MetalParser.Predicting
{
    class ARMAPredictor
    {
        public static double Predict(List<Double> values, int accuracy)
        {
            double epsilon = Normal.Sample(0.0, values.StandardDeviation);
            double X = epsilon + MAPredictor.Predict(values, accuracy);

            return X;
        }

        public static List<Double> PredictList(List<Double> values, int accuracy)
        {
            List<Double> result = new List<Double>();
            int parameter = 2;
            double[] coefficients = GetCoefficients(values, parameter);

```

```

List<Double> maModel = MAPredictor.PredictList(values, acc

for (int i = 0; i < values.Count; i++)
{
    double epsilon = Normal.Sample(0.0, values.StandardDev
    double sum = 0;
    for (int j = 0; j < parameter; j++)
    {
        if (i >= parameter)
        {
            sum += values[i - j] * coefficients[parameter
            sum += sum == 0 ? 0 : 490;
        }
        else
        {
            sum = values[i];
            break;
        }
    }
    result.Add(epsilon + (sum + maModel[i])/2.0);
}

return result;
}

private static double StandardDeviation(List<Double> values) /
{
    Double mean = values.Sum() / values.Count;
    List<Double> squaredDiff = new List<Double>(values.Count);
    for (int i = 0; i < values.Count; i++)
    {
        squaredDiff[i] = Math.Pow((values[i] - mean), 2.0);
    }
}

```

```

        return squaredDiff.Sum()/values.Count;
    }

    private static double[] GetCoefficients(List<Double> values, int parameter)
    {
        double result = 0.0;
        double[] coefficients = new double[parameter];

        //Coefficients evaluation
        for (int i = 0; i < coefficients.Count(); i++)
        {
            double upper = 0.0;
            double lower = 0.0;

            for (int j = i + 1; j < parameter; j++)
            {
                upper += (values[values.Count - j] - values.Mean())

            }
            for (int j = 0; j < parameter; j++)
            {
                lower += Math.Pow(values[values.Count - j - 1] - v

            }

            coefficients[i] = upper / lower;
        }

        return coefficients;
    }

    private static double ARModel(List<Double> values, int parameter)
    {

```

```

        double[] coefficients = GetCoefficients(values, parameter)
        double result = 0;

        for (int i = 0; i < parameter; i++)
        {
            result += values[values.Count - i - 1] * coefficients[i];
        }

        return result;
    }
}

```

SSAPredictor.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using MathNet.Numerics.LinearAlgebra;
using MathNet.Numerics.LinearAlgebra.Double;
using MathNet.Numerics.LinearAlgebra.Factorization;

namespace MetalParser.Predicting
{
    class SSAPredictor
    {
        private static int L = 150;
        private static int K;

        public static double Predict(List<Double> values, int accuracy)
        {
            return Reconstruct(values, accuracy)[0];
        }
    }
}

```

```

public static List<Double> PredictList(List<Double> values, in
{
    List<Double> orig = new List<Double>(values);
    List<Double> rec = new List<Double>(Reconstruct(values, ac

    for (int i = 0; i < values.Count; i++)
    {
        rec[i] = rec[i] * 10 + orig[i];
    }
    return rec;
}

private static Matrix<Double> BuildTrajectoryMatrix(List<Double>
{
    int N = values.Count;
    //if (L > N / 2)
    //    L = (N * 4)/10;

    K = N - L + 1;
    Matrix<Double> X = DenseMatrix.Create(K, L, 0);

    for (int i = 0; i < K; i++) //Rows
    {
        for (int j = 0; j < L; j++) //Cols
        {
            if (i + j <= values.Count - 1)
                X[i, j] = values[i + j];
        }
    }

    return X;
}

```

```

private static Matrix<Double> SVD(List<Double> values, int acc
{
    Matrix<Double> X = BuildTrajectoryMatrix(values);

    Matrix<Double> V = X.Transpose() * X;
    Svd<Double> svd = V.Svd(true);
    Matrix<Double> U = svd.VT;
    Matrix<Double> rca = U * V.Inverse();

    return rca;
}

private static Vector<Double> Reconstruct(List<Double> values,
{
    int N = values.Count;
    Matrix<Double> rca = SVD(values, accuracy);
    Vector<Double> y = DenseVector.Create(N, 0.0);
    int Lp = Math.Max(L, K);
    int Kp = Math.Min(L, K);

    for (int k = 0; k < Kp; k++)
    {
        for (int m = 1; m < k; m++)
        {
            y[k + 1] += /*(1 / (Double.Parse(k.ToString())) + 1
        }
    }

    for (int k = Lp - 1; k < Kp - 1; k++)
    {
        for (int m = 1; m < Lp; m++)
        {
            y[k + 1] += /*(1 / Double.Parse(Lp.ToString())) **

```



```

        }
    }

    for (int k = Kp; k < N; k++)
    {
        for (int m = k - Kp + 2; m < N - Lp; m++)
        {
            y[k + 1] += /*(1 / Double.Parse((N - K).ToString()
        }
    }

    return y;
}
}
}

```

Data.cs

```

using System;
using System.Runtime.Serialization;

namespace MetalParser
{
    [DataContract]
    class Data
    {
        [DataMember]
        public DateTime Date { get; set; }
        [DataMember]
        public Double Value { get; set; }

        public Data(DateTime date, Double value)
        {
            Date = date;

```

```

        Value = value;
    }
}

```

Program.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MetalParser
{
    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}

```

Form1.Designer.cs

```

namespace MetalParser
{
    partial class Form1

```

```

{
    /// <summary>
    /// Обязательная переменная конструктора.
    /// </summary>
    private System.ComponentModel.IContainer components = null;

    /// <summary>
    /// Освободить все используемые ресурсы.
    /// </summary>
    /// <param name="disposing">истинно, если управляемый ресурс д
    protected override void Dispose(bool disposing)
    {
        if (disposing && (components != null))
        {
            components.Dispose();
        }
        base.Dispose(disposing);
    }

    #region Код, автоматически созданный конструктором форм Window

    /// <summary>
    /// Требуемый метод для поддержки конструктора - не изменяйте
    /// содержимое этого метода с помощью редактора кода.
    /// </summary>
    private void InitializeComponent()
    {
        System.Windows.Forms.DataVisualization.Charting.ChartArea
        System.Windows.Forms.DataVisualization.Charting.Legend leg
        System.Windows.Forms.DataVisualization.Charting.Series ser
        this.button1 = new System.Windows.Forms.Button();
        this.button2 = new System.Windows.Forms.Button();
        this.textBox1 = new System.Windows.Forms.TextBox();
    }
}

```

```

this.label1 = new System.Windows.Forms.Label();
this.button3 = new System.Windows.Forms.Button();
this.bar = new System.Windows.Forms.DataVisualization.Charting
this.comboBox1 = new System.Windows.Forms.ComboBox();
this.label2 = new System.Windows.Forms.Label();
this.panel1 = new System.Windows.Forms.Panel();
this.MA5 = new System.Windows.Forms.Label();
this.MA10 = new System.Windows.Forms.Label();
this.MA20 = new System.Windows.Forms.Label();
this.MA50 = new System.Windows.Forms.Label();
this.MA100 = new System.Windows.Forms.Label();
this.MA200 = new System.Windows.Forms.Label();
this.button4 = new System.Windows.Forms.Button();
((System.ComponentModel.ISupportInitialize)(this.bar)).Begin
this.panel1.SuspendLayout();
this.SuspendLayout();
//
// button1
//
this.button1.Location = new System.Drawing.Point(12, 33);
this.button1.Name = "button1";
this.button1.Size = new System.Drawing.Size(75, 23);
this.button1.TabIndex = 1;
this.button1.Text = "Начать";
this.button1.UseVisualStyleBackColor = true;
this.button1.Click += new System.EventHandler(this.button1
//
// button2
//
this.button2.Enabled = false;
this.button2.Location = new System.Drawing.Point(94, 32);
this.button2.Name = "button2";
this.button2.Size = new System.Drawing.Size(75, 23);

```

```

this.button2.TabIndex = 2;
this.button2.Text = "Закончить";
this.button2.UseVisualStyleBackColor = true;
this.button2.Click += new System.EventHandler(this.button2
//
// textBox1
//
this.textBox1.AcceptsReturn = true;
this.textBox1.AcceptsTab = true;
this.textBox1.Anchor = ((System.Windows.Forms.AnchorStyles
| System.Windows.Forms.AnchorStyles.Left));
this.textBox1.ImeMode = System.Windows.Forms.ImeMode.On;
this.textBox1.Location = new System.Drawing.Point(13, 81);
this.textBox1.Multiline = true;
this.textBox1.Name = "textBox1";
this.textBox1.ReadOnly = true;
this.textBox1.Size = new System.Drawing.Size(258, 335);
this.textBox1.TabIndex = 3;
this.textBox1.TextChanged += new System.EventHandler(this.
//
// label1
//
this.label1.AutoSize = true;
this.label1.Location = new System.Drawing.Point(10, 65);
this.label1.Name = "label1";
this.label1.Size = new System.Drawing.Size(51, 13);
this.label1.TabIndex = 4;
this.label1.Text = "Samsung";
this.label1.Click += new System.EventHandler(this.label1_C
//
// button3
//
this.button3.Location = new System.Drawing.Point(277, 55);

```

```

this.button3.Name = "button3";
this.button3.Size = new System.Drawing.Size(121, 23);
this.button3.TabIndex = 19;
this.button3.Text = "Построить график";
this.button3.UseVisualStyleBackColor = true;
this.button3.Click += new System.EventHandler(this.button3
//
// bar
//
chartArea2.Name = "ChartArea1";
this.bar.ChartAreas.Add(chartArea2);
legend2.Name = "Legend1";
this.bar.Legends.Add(legend2);
this.bar.Location = new System.Drawing.Point(277, 81);
this.bar.Name = "bar";
this.bar.Palette = System.Windows.Forms.DataVisualization.
series2.ChartArea = "ChartArea1";
series2.Legend = "Legend1";
series2.Name = "Time Series";
this.bar.Series.Add(series2);
this.bar.Size = new System.Drawing.Size(630, 335);
this.bar.TabIndex = 20;
this.bar.Text = "chart1";
//
// comboBox1
//
this.comboBox1.Cursor = System.Windows.Forms.Cursors.Hand;
this.comboBox1.FormattingEnabled = true;
this.comboBox1.Items.AddRange(new object[] {
"MA",
"ARMA",
"SSA"});
this.comboBox1.Location = new System.Drawing.Point(405, 55

```

```

this.comboBox1.Name = "comboBox1";
this.comboBox1.Size = new System.Drawing.Size(121, 21);
this.comboBox1.TabIndex = 21;
this.comboBox1.SelectedIndexChanged += new System.EventHandler
//
// label2
//
this.label2.AutoSize = true;
this.label2.Location = new System.Drawing.Point(402, 38);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(108, 13);
this.label2.TabIndex = 22;
this.label2.Text = "Модель построения";
//
// panel1
//
this.panel1.Controls.Add(this.MA200);
this.panel1.Controls.Add(this.MA100);
this.panel1.Controls.Add(this.MA50);
this.panel1.Controls.Add(this.MA20);
this.panel1.Controls.Add(this.MA10);
this.panel1.Controls.Add(this.MA5);
this.panel1.Location = new System.Drawing.Point(620, 3);
this.panel1.Name = "panel1";
this.panel1.Size = new System.Drawing.Size(260, 75);
this.panel1.TabIndex = 23;
this.panel1.Visible = false;
//
// MA5
//
this.MA5.AutoSize = true;
this.MA5.Location = new System.Drawing.Point(4, 4);
this.MA5.Name = "MA5";

```

```

this.MA5.Size = new System.Drawing.Size(29, 13);
this.MA5.TabIndex = 0;
this.MA5.Text = "MA5";
//
// MA10
//
this.MA10.AutoSize = true;
this.MA10.Location = new System.Drawing.Point(4, 29);
this.MA10.Name = "MA10";
this.MA10.Size = new System.Drawing.Size(35, 13);
this.MA10.TabIndex = 1;
this.MA10.Text = "MA10";
//
// MA20
//
this.MA20.AutoSize = true;
this.MA20.Location = new System.Drawing.Point(4, 52);
this.MA20.Name = "MA20";
this.MA20.Size = new System.Drawing.Size(35, 13);
this.MA20.TabIndex = 2;
this.MA20.Text = "MA20";
//
// MA50
//
this.MA50.AutoSize = true;
this.MA50.Location = new System.Drawing.Point(137, 4);
this.MA50.Name = "MA50";
this.MA50.Size = new System.Drawing.Size(35, 13);
this.MA50.TabIndex = 3;
this.MA50.Text = "MA50";
//
// MA100
//

```



```

this.MA100.AutoSize = true;
this.MA100.Location = new System.Drawing.Point(137, 29);
this.MA100.Name = "MA100";
this.MA100.Size = new System.Drawing.Size(41, 13);
this.MA100.TabIndex = 4;
this.MA100.Text = "MA100";
//
// MA200
//
this.MA200.AutoSize = true;
this.MA200.Location = new System.Drawing.Point(137, 52);
this.MA200.Name = "MA200";
this.MA200.Size = new System.Drawing.Size(41, 13);
this.MA200.TabIndex = 5;
this.MA200.Text = "MA200";
//
// button4
//
this.button4.Location = new System.Drawing.Point(536, 12);
this.button4.Name = "button4";
this.button4.Size = new System.Drawing.Size(82, 23);
this.button4.TabIndex = 24;
this.button4.Text = "Прогноз";
this.button4.UseVisualStyleBackColor = true;
this.button4.Click += new System.EventHandler(this.Button4
//
// Form1
//
this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
this.BackColor = System.Drawing.SystemColors.GradientInactiveCaption;
this.ClientSize = new System.Drawing.Size(919, 428);
this.Controls.Add(this.button4);

```

```

        this.Controls.Add(this.panel1);
        this.Controls.Add(this.label2);
        this.Controls.Add(this.comboBox1);
        this.Controls.Add(this.bar);
        this.Controls.Add(this.button3);
        this.Controls.Add(this.label1);
        this.Controls.Add(this.textBox1);
        this.Controls.Add(this.button2);
        this.Controls.Add(this.button1);
        this.Name = "Form1";
        this.Text = "Form1";
        this.Shown += new System.EventHandler(this.Form1_Load);
        ((System.ComponentModel.ISupportInitialize)(this.bar)).EndInit();
        this.panel1.ResumeLayout(false);
        this.panel1.PerformLayout();
        this.ResumeLayout(false);
        this.PerformLayout();

    }

```

#endregion

```

private System.Windows.Forms.Button button1;
private System.Windows.Forms.Button button2;
private System.Windows.Forms.TextBox textBox1;
private System.Windows.Forms.Label label1;
private System.Windows.Forms.Button button3;
private System.Windows.Forms.DataVisualization.Charting.Chart chart1;
private System.Windows.Forms.ComboBox comboBox1;
private System.Windows.Forms.Label label2;
private System.Windows.Forms.Panel panel1;
private System.Windows.Forms.Label MA200;
private System.Windows.Forms.Label MA100;
private System.Windows.Forms.Label MA50;

```

```
private System.Windows.Forms.Label MA20;  
private System.Windows.Forms.Label MA10;  
private System.Windows.Forms.Label MA5;  
private System.Windows.Forms.Button button4;  
}  
}
```