**Тиамат**  
Интелигентна система за анализ на финансови пазари и генериране на търговски сигнали.

## АВТОР:

Име: Антон Дончев Донев

Адрес: гр. Казанлък, ул. „Тулово“ №2

E-mail:  antondonevv@gmail.com

Телефон:  0896342700

ЕГН:  0645017624

Училище: ППМГ „Никола Обрешков“, гр. Казанлък

Клас: 12 б

## РЪКОВОДИТЕЛ:

  Име: Здравка Стефанова Димитрова

  Телефон: +359893422519

  E-mail: [dimitrova@pmgkk.com](mailto:dimitrova@pmgkk.com)

  Длъжност: учител по информатика и информационни технологии

## РЕЗЮМЕ:

# 3.1 Цели

### 3.1.1 Предназначение

В 9 клас започнах да проявавам сериозен интерес към пазарите и икономиката, което ме вдъхнови да изучавам задълбочено пазарни принципи, математическа вероятност и стратегии за анализ. Паралелно с това, като ученик в паралелка с интензивно обучение на програмиране, се занимавах със софтуери и кодиране. След натрупването на известен практически опит на пазарите и в софтуерната област, осъзнах, че съществува огромен потенциал за комбинирането на тези две сфери – финансите и изкуствения интелект (ИИ).

Понастоящем много хора са скептични за бъдещето на професията „програмист“, считано, че моделите на изкуствения интелект могат да заместят софтуерните специалисти. Вярвам, че подобен скептицизъм е неоправдан, особено преди епохата на т.нар. „AGI“ (Artificial General Intelligence). На настоящия етап AI е отличен помощник и инструмент, който може да автоматизира рутинни задачи и да надгради възможностите на един професионалист, но все още не притежава универсална човешка интелигентност и креативност.

Затова реших да развия проект, който съчетава натрупания ми финансов опит и уменията ми по програмиране. Моята цел е да създам сложен модел, способен да анализира големи обеми пазарни данни и да генерира сигнали, които да „победят“ пазара. Този проект, озаглавен **„Тиамат“**, онагледява как един човек, въоръжен с достатъчно усилия, знания и подкрепата на изкуствения интелект, може да изгради иновативно решение в мащаб, който до скоро се считаше за непостижим извън големите финансови институции или хедж фондове.

## 3.2 Основни етапи в реализиране на проекта

# 3.2.1 Основни дейности при създаването на проекта „Тиамат“

1. **Провеждане на задълбочено проучване**
   * Източници на пазарна информация:
     + Избор на платформа за събиране на исторически данни – в случая *MetaTrader* с брокера *IC Markets*.
     + Проучване на достъпни безплатни или частично платени бази данни, които могат да предоставят пазарна информация.
   * Анализ на наличните данни и техните особености:
     + Как да се справя с липсващи или непълни данни и дали тези пропуски ще повлияят сериозно на модела?
2. **Избор на модел за машинно обучение**
   * Поради факта, че достъпът до напълно изчистени и висококачествени (платени) пазарни данни е ограничен, бе предпочетен **LightGBM**.
     + Предимства:
       - Бързо обучение и ниско използване на ресурси.
       - Добре се справя с липсващи данни и не толкова „изчистени“ набори.
       - Подходящ за работа с големи обеми (милиони) записи.
   * Обмисляне на алтернативни модели (невронни мрежи, случайни гори и др.) и причини да бъдат отхвърлени (например прекалено дълго обучение, нужда от много висококачествени данни, сложна архитектура).
3. **Обработка и трансформация на данните**
   * **Филтриране** и **нормализация**:
     + Как да направя така, че неточни, дублирани или липсващи стойности да не изкривят резултатите?
   * **Изчисляване на технически индикатори**:
     + Определяне кои индикатори (Moving Averages, RSI, MACD и др.) са най-подходящи, без да се стига до „претоварване“ на модела с ненужни данни.
     + Премахване на силно корелиращи индикатори, за да се избегне излишна изчислителна сложност и „overfit“.
   * **Изключване на периоди с големи новини** (като решения на Федералния резерв или други събития с потенциално голямо влияние на цената) или специална обработка на тези периоди, за да не подвеждат модела.
4. **Избор и проектиране на база данни**
   * Изисквания:
     + Висока скорост на запис и четене, особено за времеви серии (minute-level data).
     + Лесно мащабиране и работа с милиони редове.
   * **QuestDB** като решение:
     + Специализирана в обработката на времеви серии.
     + Позволява бързи заявки, търсене и анализ върху големи обеми данни.
     + Предоставя ефективен начин за *„seeding“* (първоначално зареждане) на данните, нужни за обучение и валидация на модела.
5. **Разработване и обучение на AI модела**
   * **Сглобяване на обучителния набор**:
     + Експортиране на данни от QuestDB и структуриране във формат, подходящ за LightGBM.
     + Обхващане на дълъг исторически период (2018–2025) с възможно най-малки пропуски.
   * **Конфигуриране на LightGBM**:
     + Настройка на хиперпараметри (learning rate, num\_leaves и др.) с цел постигане на баланс между точност и бързина на обучение.
     + Използване на методи за избягване на overfit (cross-validation и други).
   * **Валидация**:
     + Разделяне на данните на тренировъчен, валидационен и тестов сет.
     + Анализ на метрики (точност, прецизност, загуба, печалба, стигната цел и други).
6. **Определяне на стратегия за вход и изход от позиции**
   * **Статична** vs. **динамична** стратегия:
     + Статична: твърди правила с определени стоп лос и тейк профит нива.
     + Динамична: AI моделът адаптивно генерира сигнали на база на обучението си и текуща пазарна ситуация, без твърди фиксирани нива.
   * Преценка дали да се включи информация за „sentiment“:
     + Липсата на платени ресурси за анализ на социални мрежи/новини ограничава модела единствено до технически данни, но това може да се окаже достатъчно за определени пазари.
7. **Интеграция и оптимизация**
   * **Хостинг** и изпълнение на модела:
     + Избор на подходяща облачна услуга (например AWS, Google Cloud или Azure) или локален сървър.
   * **Оптимизация на производителността**:
     + Проследяване на времето за зареждане и обработка на големите исторически масиви от данни.
     + Оптимална структура на таблиците в QuestDB, индекси и начина на заявките.
     + Подбор на подходящ хардуер (RAM, CPU, GPU) спрямо нуждите на модела.
8. **Тестване и валидиране на крайния прототип**
   * **Симулация на реална търговия** (backtesting) на исторически данни, за да се прецени каква доходност би реализирал моделът при реални условия.
   * **Проверка на реакция при пазарни шокове**:
     + Рязка смяна в пазарни условия (Fed новини, политически събития и др.).
     + Анализ дали моделът успява да избегне сериозни „slippage“ моменти.
   * **Paper Trading** (forward testing) с демо акаунт:
     + Тестване в реално време, но без реален риск на средства.
     + Събиране на допълнителни данни за бъдещо подобрение.

## 3.2 Ниво на сложност

Проектът „Тиамат“ има висока степен на сложност поради няколко ключови фактора:

1. **Анализ на големи обеми данни (Big Data)**
   * Системата обработва минутни ценови данни от пазари с няколко годишна давност (2018–2025), което води до милиони записи в базата. Това изисква ефективно структуриране, филтриране и бърз достъп до данните.
2. **Специфика на интеграцията с MetaTrader**
   * Необходимо е да се създаде скрипт (експертен съветник, EA), написан на MQL (подобен на C език), който да получава и обработва генерираните от ИИ сигнали.
   * Допълнителна сложност внася нуждата от „мост“ между Python кода (ИИ) и MetaTrader, реализиран чрез DLL библиотека. Python модулът отваря локален порт, а DLL библиотеката на MetaTrader комуникира с този порт, предавайки търговските сигнали и данни в реално време.
   * Осигуряването на сигурна комуникация изисква допълнителни мерки, като **филтриране на IP адреси** (списък с позволени IP адреси), така че да се предотврати неоторизиран достъп.
3. **Софтуерна архитектура и инфраструктура**
   * Едновременното използване на специализирана база за времеви серии (QuestDB), интеграция с MetaTrader (чрез EA + DLL) и Python логика за машинно обучение усложнява дизайна на системата и поддръжката ѝ.
4. **Комплексност на машинното обучение**
   * Разработването и тренирането на модела (LightGBM) върху непълни, „шумни“ изисква прецизна обработка, хиперпараметрична оптимизация и техники за предпазване от пренасищане (overfitting).
   * Системата трябва да работи в реално време и да подава сигнали своевременно, без да забавя търговските операции на MetaTrader.
5. **Динамични пазарни условия**
   * Финансовите пазари са силно непредвидими, като съществуват екстремни събития (новини, политически фактори, решения на Федералния резерв и др.). Интегрирането на логика за разпознаване/филтриране на подобни събития увеличава сложността на алгоритмите и бизнес логиката.
6. **Високи изисквания към производителността**
   * За да се генерират навременни сигнали и да се отварят/затварят позиции в реално време, системата трябва да гарантира ниска латентност при зареждане и анализ на данни. Това налага специализирани оптимизации при обработката и заявките към базата (QuestDB), както и ефективен дизайн на комуникационния „мост“ (Python ↔ DLL ↔ MetaTrader).
7. **Backtesting и forward testing**
   * Симулацията на историческо тестване (backtesting) за продължителни периоди от време и обемисти данни е само по себе си предизвикателна. Необходимо е изграждането на механизми, които коректно да пресъздават пазарните условия.
   * Наличието на „paper trading“ (forward testing) в реално време добавя още едно ниво на интеграция, контрол и мониторинг.
8. **Богата бизнес логика и стратегии**
   * Системата поддържа гъвкавост при определяне на стратегии – от статична (фиксирани стоп лос/тейк профит) до динамична (AI-базирани сигнали, адаптиращи се спрямо пазарните условия). Това усложнява управлението на различни търговски сценарии и логики за реакция при промяна на пазарния тренд.

Всички тези аспекти изискват задълбочени познания в няколко направления: програмиране (Python, MQL, DLL), бази данни за времеви серии, статистика, машинно обучение и финансов инженеринг. Многослойната структура на „Тиамат“ трябва да бъде едновременно бърза, надеждна и сигурна, за да предоставя предимство в силно конкурентната и непрекъснато променяща се търговска среда.

## ****4.4 Логическо и функционално описание на решението****

„Тиамат“ е многокомпонентна система, която обединява няколко ключови функционални модула и специфична логика при обмена на данни и генерирането на търговски сигнали. По-долу е описана основната архитектура, както и отделните роли и функции на елементите, които вземат участие в процеса.

# 1. Модули в системата

1. **Модул за събиране и съхранение на данни**
   * **Функция:** Събира исторически и текущи пазарни данни (цени, обеми и др.) от избран източник (MetaTrader / външни бази).
   * **Инструменти:**
     + MetaTrader (MQL EA / скрипт): Получава поточни котировки и ги препраща към базата.
     + QuestDB: Основна база данни за времеви серии, в която записите се индексират за бърз достъп и ефективно управление на големи обеми данни.
   * **Логика:** MetaTrader предоставя архиви с пазарните данни и ги подава към QuestDB. Базата се грижи за надеждното и бързо съхранение, за да може по-късно тези данни да се използват при обучението на модела.
2. **Модул за предобработка и трансформация**
   * **Функция:** Филтрира, нормализира и обогатява базовите данни с технически индикатори (Moving Averages, RSI, MACD и др.).
   * **Инструменти:**
     + Python скриптове: Използват данните от QuestDB, прилагат методи за почистване (премахване на дублиращи стойности, справяне с липсващи данни) и изчисляват технически индикатори.
   * **Логика:** Суровите данни (цени, обеми, времеви отметки) се обработват, за да се генерира оптимален набор от характеристики (features), които да бъдат подадени към алгоритъма за машинно обучение.
3. **Модул за машинно обучение (LightGBM)**
   * **Функция:** Анализира подготвените данни, научава модел за прогноза на пазара и генерира сигнали за вход/изход от позиции.
   * **Инструменти:**
     + LightGBM: Библиотека за градиентно буустинг, особено подходяща за големи и частично „шумни“ набори от данни.
   * **Логика:**
     + **Обучение:** Историческите данни (разделени на тренировъчен, валидационен и тестов сет) се използват за настройка на модела. Прилагат се техники за избягване на пренасищане (overfitting) и оптимизиране на хиперпараметрите.
     + **Предсказване:** В реално време (или на определени интервали) моделът получава текущите пазарни условия (индикатори, цени) и връща препоръки за покупка/продажба създадени чрез вероятности и статистика.
4. **Модул за търговско изпълнение и мост (Python ↔ DLL ↔ MetaTrader)**
   * **Функция:** Предава генерираните от LightGBM сигнали към MetaTrader, където се изпълняват реални (или демо) сделки.
   * **Инструменти и логика:**
     + **DLL библиотека:**
       - Написана на C/C++ и интегрирана с MetaTrader (MQL EA), осигурява постоянна TCP връзка към Python сървъра.
       - Получава сигнали от Python и ги съхранява във временна променлива, достъпна от EA при всяко ново запитване (или нов тик).
     + **Python сървър:**
       - Слуша на локален (или външен) порт и обработва заявки само от доверени IP адреси (филтър на IP-тата).
       - При поискване изпраща актуален сигнал (BUY/SELL) или други данни, генерирани от модела, към DLL библиотеката.
     + **MQL EA (експертен съветник):**
       - Периодично (при всеки тик) извиква функция от DLL и проверява дали има нов сигнал.
       - При наличие на сигнал, EA може да стартира сделка (BUY/SELL), да затвори позиция или да модифицира стоп лос, следвайки предварително зададената логика.
5. **Модул за тестване и оптимизация**
   * **Функция:** Позволява извършване на backtesting (исторически тест) и forward testing („paper trading“) за оценка на ефективността на модела, преди да се използват реални средства.
   * **Логика:**
     + **Backtesting:** Като се използват историческите данни от QuestDB, Python скриптовете емулират минали пазарни условия и подават съответните цени и индикатори към модела. Така се изчислява дали стратегията би била печеливша.
     + **Forward testing (Paper Trading):** Моделът подава сигнали в реално време към MetaTrader EA, но сделките се изпълняват с виртуален баланс, за да може трейдърът да оцени реалния потенциал без риск.

# 2. Взаимодействие между компонентите

1. **MetaTrader (MQL EA) ↔ DLL библиотека**
   * EA периодично (обикновено при всеки нов пазарен тик) извиква функция от DLL, за да изчете дали има нов сигнал („BUY“, „SELL“ или друг тип команда).
   * DLL, от своя страна, поддържа непрекъсната връзка към Python сървъра и при нужда съобщава новопристигналите сигнали на EA.
2. **DLL библиотека ↔ Python сървър**
   * DLL се свързва по TCP с Python сървъра, като при отпадане на връзката периодично опитва да се свърже отново (напр. на всеки 60 секунди).
   * Python сървърът приема заявки само от предварително разрешени IP адреси и след валидация връща сигнали или друга необходима информация (например текуща препоръка, цена, статус на модел и т.н.).
3. **Python сървър ↔ QuestDB**
   * След получаване на резултатите и евентуалното генериране на нови сигнали, сървърът ги предоставя към DLL, откъдето стигат до MetaTrader.

С това се осигурява цялостен цикъл – от събирането на данни през анализа с машинно обучение, до автоматизираното изпълнение на сделки в реално или демо време, осигурявайки висока степен на автоматизация и гъвкавост на финансовите търговски операции.

## Данните се използват, за да се въведе текущият пазарен контекст на модела, както и за натрупване на статистика при тестване.

1. **Обучение на модела**
   * Извършва се офлайн върху голям набор исторически данни от QuestDB.
   * След като моделът бъде обучен, новите му параметри се зареждат в Python сървъра, който вече използва актуализирания модел за прогнози.
2. **Backtesting и Forward Testing**
   * **Backtesting** – Python скриптовете емулират минали периоди и подават данните към модела, за да се създаде хронология от сделки и да се изчисли печалбата, процента печеливши сделки, максималния спад (drawdown) и др.
   * **Forward Testing (Paper Trading)** – MetaTrader EA отваря/затваря позиции в демо среда, докато DLL и Python сървърът продължават да осигуряват сигнали в реално време.

(Ще бъде поставен речник накрая на документа, за по-лесно разбиране на терминологията)

**Речник на понятията (Терминология)**

1. **Backtesting (бек-тестване)**  
   Метод за оценяване на ефективността на дадена търговска стратегия или AI модел чрез прилагане на стратегията върху исторически данни. Целта е да се види как би се представил моделът в минал период, преди да се използва в реално време.
2. **Forward testing (paper trading)**  
   Извършване на тестове върху стратегията или модела в реално време, но без да се поставят реални парични средства на риск. Най-често се прави в демо акаунт при брокер, за да се видят резултатите от сделките при актуални пазарни условия.
3. **LightGBM**  
   Библиотека за машинно обучение, базирана на градиентни буустинг алгоритми (Gradient Boosting Machine). Характеризира се с висока скорост на обучение, ниска консумация на памет и отлична способност за справяне с големи набори от данни и липсваща информация.
4. **Sentiment (сентимент)**  
   Фактор, отнасящ се до емоционалната или психологическата реакция на пазара – нагласата на масовите участници (трейдъри, инвеститори) към даден актив или пазарна ситуация. Обикновено се извлича от анализ на новини, форуми, социални мрежи и др.
5. **Slippage (слипидж)**  
   Разликата между очакваната цена за изпълнение на дадена поръчка и реалната цена, на която поръчката се изпълнява. Най-често се появява при периоди на висока волатилност или ниска ликвидност, когато цената се променя бързо.
6. **Stop Loss (стоп лос)**  
   Ниво или цена, при която трейдърът желае да приключи загубна позиция, за да ограничи възможните щети. Ако пазарът достигне това ниво, позицията се затваря автоматично.
7. **Take Profit (тейк профит)**  
   Ниво или цена, при която трейдърът предпочита да затвори печеливша позиция, за да реализира печалбата си. При достигане на това ниво, позицията се затваря автоматично.
8. **Cross-Validation (крос-валидация)**  
   Техники в машинното обучение за по-надеждна оценка на модела. Данните се разделят на няколко поднабора (folds), като последователно един от подмножества се ползва за тестване, а останалите – за обучение. Така се намалява рискът от „overfitting“ (прекалено нагаждане към тренировъчния набор).
9. **Overfit (преобучаване)**  
   Състояние, при което моделът научава специфични (често шумови) особености на тренировъчните данни, но не се представя добре върху нови, невидяни досега данни. В резултат прогнозите извън тренировъчния набор са неточни.
10. **Market Data (пазарни данни)**  
    Информацията, свързана с движението на цените на активите – акции, валутни курсове (Forex), криптовалути и т.н. Обикновено се състои от цена на отваряне, затваряне, най-висока, най-ниска цена (OHLC), както и обем на търговия (Volume) и други метаданни.
11. **QuestDB**  
    Високопроизводителна база данни, оптимизирана за времеви серии (Time Series Database), позволяваща ефективно съхранение и работа с големи обеми от хронологично подредени записи (примерно котировки за всека минута, исторически данни и др.).