**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО»**
**Інститут прикладного системного аналізу**
**Кафедра математичних методів системного аналізу**

«До захисту допущено»
В. о. завідувача кафедри
_____ О.Л. Тимощук
«___»_____20__ р.

**Дипломна робота**

**на здобуття ступеня бакалавра**
**з напряму підготовки 6.050101 «Комп'ютерні науки»**
**на тему: «Моделі нелінійних нестаціонарних процесів у фінансах»**

Виконав:
студент IV курсу, групи КА-55
Гуць Євгеній Віталійович

_____
Керівник:
професор, д.т.н. Бідюк П.І.

_____
Консультант з економічного розділу:
доцент, к.е.н. Шевчук О.А. _____

Консультант з нормоконтролю:
доцент, к.т.н. Коваленко А. Є. _____

Рецензент:
проф., д. т. н. Теленик С.Ф. _____

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студент _____

Київ – 2019 року

# РЕФЕРАТ

Дипломна робота: 101 с., 19 рис., 14 табл., 2 додатки, 15 джерел.

НЕЛІНІЙНІ НЕСТАЦІОНАРНІ ПРОЦЕСИ, ФІНАНСОВІ ПРОЦЕСИ, ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ, БАЙЄСІВСЬКІ МЕРЕЖІ, ПРОГНОЗУВАННЯ

Дана робота присвячена вивченню методів прогнозування та методик побудови моделей нелінійних нестаціонарниї процесів, а саме методу байєсівських мереж та застосування його на статистичних наборах фінансових даних.

Метою дипломної роботи є вивчення основних засад побудови моделі та прогнозування з використанням байєсівських мереж, а також розробка програмного продукту для отримання практичних результатів та порівняння роботи алгоритму на різних наборах фінансових даних.

Об’єктом дослідження нелінійні нестаціонарні процеси, подані у вигляді статистичних даних стосовно позичальників кредитів, які використовуються для побудови ймовірнісно-статистичних моделей у формі байєсівських мереж.

# ABSTRACT

The work consist of 101 pages, 19 images, 14 tables , 2 appendices, 15 sources.

The theme: «Models of non-linear non-stationary processes in finances».

NONLINEAR PROCESSES, FINANCIAL PROCESSES, INTELLECTUAL ANALYSIS OF DATA, BAYESNIC NETWORKS, FORECASTING

This work is devoted to the study of forecasting methods and methods of constructing models of nonlinear non-stationary processes, the Bayesian network method and its application on statistical data sets of financial data.

The purpose of the thesis is to study the main principles of model construction and forecasting using Bayesian networks, as well as the development of a software product for obtaining practical results and comparing the operation of the algorithm on different sets of financial data.

The object of the work is nonlinear non-stationary processes, presented in the form of statistical data on borrowers, used to construct probabilistic-statistical models in the form of Bayesian networks.

# ЗМІСТ

# ВСТУП

Фінансова діяльність та пов’язані з нею економічні процеси стали невід’ємною частиною нашої буденності та супроводжують на кожному кроці нашого повсякденного життя. Тому їх вивчення, а саме дослідження нелінійних нестаціонарних процесів, вимагає все більшої уваги аналітиків.

Сьогодні ми спостерігаємо накопичення великих обсягів інформації, які потребують відповідної обробки для прийняття певних рішень. Проблеми математичного моделювання стають все більше актуальними для ефективної організації управління певними суб'єктами господарювання та економічними спільнотами через те, що якість прийнятих рішень значною мірою залежить від якості прогнозування їх наслідків. Саме тому рішення, прийняті сьогодні, повинні грунтуватися на достовірних оцінках можливого розвитку процесів та подій в майбутньому.

Для пошуку закономірностей в масивах даних використовуються методи інтелектуального аналізу даних. Вони будуються на використанні однієї з двох технології: машинного навчання (Machine Learning) чи візуалізації (візуальне подання інформації). Байєсівські мережі цікаві тим, що об’єднують у собі одразу дві технології.

Байєсові мережі є типом ймовірнісної графічної моделі, яка використовує байєсівський висновок для обчислення ймовірностей. Байєсові мережі прагнуть моделювати умовну залежність і, отже, причинно-наслідковий зв'язок, подаючи умовну залежність ребрами в орієнтованому графі.

Байєсові мережі широко використовуються в медичній і інженерній діагностиці при неповній і недосконалій інформації, в системах класифікації даних, автоматичному розпізнаванні мови, маркетингу, бізнесі і багатьох

інших заходах. У загальному випадку BN дозволяють відтворювати причинно-наслідкові зв'язки між подіями і визначати ймовірність події, якщо нова інформація про зміни стану будь-якого вузла (змінної) мережі стають доступними. Ступінь доцільності застосування цього методу моделювання та ймовірнісного висновку залежить від здатності правильно сформулювати задачу, відібрати змінні процесу, що характеризують її динаміку або статику в достатній мірі, знайти необхідні дані та використовувати їх для мережевого навчання. і правильно сформулювати результуючий висновок за допомогою побудованої мережі.

# РОЗДІЛ 1 ОСОБЛИВОСТІ РОЗВИТКУ ФІНАНСОВИХ ПРОЦЕСІВ ТА ПОБУДОВИ ЇХ МАТЕМАТИЧНИХ МОДЕЛЕЙ

## 1.1 Актуальність задачі

Створення моделей та прогнозування відіграє важливу роль у прийнятті рішень. Під прогнозуванням розуміється передбачення майбутнього стану тієї чи іншої системи, спираючись на накопичений минулий досвід і припущення стосовно майбутнього.

Багато банківських установ використовують власні системи для прогнозування ризиків при кредитуванні, проте більшість таких систем не завжди роблять правильний прогноз, що, зважаючи на великі грошові потоки, може перерости у серйозні збитки.

Прогнозування відіграє досить важливу роль у роботі будь-якого підприємства, адже коли ми можемо приблизно оцінити, що очікує нас у близькому майбутньому, можна змінити напрямок розвитку підприємства, методи, які використовуються, інколи і вберегти від банкрутства.

Вимоги сучасного бізнесу змушують переходити від простого статистичного аналізу до більш складного інтелектуального аналіз даних. Він має забезпечити побудову моделі для опису інформації і в кінцевому результаті призвести до необхідних висновків.

Інтелектуальним аналізом даних називається процес обробки інформації та виявлення в ній ключових ознак, які допомагають приймати рішення. Основні принципи інтелектуального аналіз даних відомі досить давно, проте з появою досить об’ємних масивів даних вони набули особливої актуальності. Ці масиви даних призвели до зростання популярності методів інтелектуального аналізу даних, зокрема, тому, що крім великих обсягів, інформація стає ще більш різноманітною і всеохоплюючою.

Процес аналізу даних та побудови моделі здебільшого є ітеративним, оскільки потрібно відшукати та виявити різні відомості. Необхідно також розуміти як поєднати їх з іншими даними для отримання результату.

## 1.2 Аналіз існуючих підходів обробки фінансових даних

Алгоритм в інтелектуальному аналізі даних – це набір евристики та обчислень, який на основі цих даних створює модель. Алгоритм працює таким чином: спочатку аналізує наявні дані, а потім здійснює пошук певних закономірностей. Результат цього аналізу використовується ітеративно для створення оптимальних параметрів моделі інтелектуального аналізу даних. Згодом ці параметри використовуються до всього набору даних з метою виявлення придатних до використання закономірностей.

Можна видалити наступні форми моделей:

– набір кластерів, що описують зв’язки варіантів з набору даних;

– дерево рішень, яке передбачає результат і описує вплив на нього різних факторів;

– математична модель, що здійснює прогнозування;

– набір правил, що описують групування об’єктів в транзакціях та їхні ймовірності.

Здебільшого вибір правильного алгоритму для використання в певній аналітичній задачі є досить непростим завданням. Можлива ситуація, коли використання різних алгоритмів для однієї і тієї ж задачі призведе до отримання різних результатів, крім того, деякі алгоритми можуть видавати більше одного типу результатів.

Наприклад, алгоритм дерева прийняття рішень можна використовувати не лише для прогнозування, а також для зменшення кількості стовпців в наборі даних, оскільки він може виявляти стовпці, що не впливають на кінцеву модель інтелектуального аналізу даних.

### 1.2.1 Методи інтелектуального аналізу даних

А) Класифікація

Основна задача методу – встановити, до якого з класів відносяться ті чи інші дані, при цьому множина класів має бути заздалегідь визначена. Кожен клас володіє певними властивостями, що характеризують його об’єкти. Наприклад, в задачі виявлення кредитоспроможності клієнта банківський працівник оперує двома класами: «кредитоспроможний» і «некредитоспроможний». Віднести клієнта до однієї із груп допомагає аналіз ряду характеристик: віку, рівня доходів, місця роботи та інші. Це означає, що задача інтелектуального аналізу даних зводиться до того, щоб визначити один із параметрів об’єкта, аналізуючи значення інших його параметрів. На мові математики це звучить так: визначається значення залежної змінної кредитоспроможність, яка може набувати значення так або ні за відомими значеннями незалежних змінних (вік, рівень доходу, місце роботи та інші).

Для класифікації в Data Mining використовуються нейронні мережі, дерева рішень, опорні вектори, метод к-середніх, алгоритми покриття та інші.

Б) Кластеризація

Метод зовні дуже схожий на класифікацію. По суті є логічним його продовженням, узагальненням задачі класифікації, коли набір класів наперед невідомий. Цей метод ще називають кластерним аналізом. Кластеризація

відрізняється від класифікації тим, що для здійснення аналізу не вимагається мати окрему залежну змінну. Задачі розв’язується на початкових етапах дослідження, коли про дані мало що відомо. Її розв’язання допомагає краще усвідомити дані і в цьому сенсі задача кластеризації є описовою. Після визначення кластерів застосовуються інші методи інтелектуального аналізу з метою встановлення, що означає таке розбиття і чим воно зумовлене.

Наприклад, цим методом користуються навіть діти, відрізняючи об’єкти за принципом схожості-несхожості. Саме так вони відрізняють круг від квадрата, чоловіків від жінок, кішок від собак. Ішим прикладом є задача сегментації ринку, в основі якої лежить припущення, що всі клієнти різні, проте їх можна за схожістю розділити на певну кількість груп.

В) Прогнозування

Цим методом часто користуються бізнесмени, аналізуючи відомі дані, що дає можливість побудови прогнозу на майбутнє. До того ж чим детальніші історичні дані і чим більший часовий проміжок, ти точнішими виявляються результати. Цей метод часто використовується для передбачення потреби в кадрах, оцінки попиту на товари та послуги, прогнозування ринку збуту продукції. Якщо, наприклад, власник квіткового магазину хоче визначити скільки квітів потрібно замовити на 8 Березня, він має проаналізувати цифри відповідних продажів за останні 5 років.

### 1.2.2 Основні етапи інтелектуального аналізу даних

Аналітики виділяють наступні етапи інтелектуального аналіз даних:

А) Розуміння бізнесу: визначення мети проекту і вимог з боку бізнесу. Згодом ці знання втілюються в постановку задачі і попередній план по досягненню мети проекту.

Б) Розуміння даних: починається із збору даних і має на меті їх освоєння. Для цього необхідно виявити проблеми з якістю даних і сформулювати гіпотези про приховані закономірності цих даних.

В) Підготовка даних: метою є отримання результуючих наборів даних, що будуть використовуватись при моделюванні з різноформатних даних. При цьому відбувається відбір таблиць, записів та атрибутів, а також конвертація і очистка даних для моделювання.

Г) Моделювання: до даних застосовуються різноманітні методики, будуються моделі, їх параметри оптимізуються.

Д) Оцінка: передбачає оцінювання якості моделі перед її втіленням та виявлення бізнес-завдань, що були недостатньо вивчені.

Е) Розгортання: фінальний етап, що полягає у створені звіту, або автоматизації процесу аналізу даних для вирішення бізнес-задач.

## 1.3 Особливості розвитку нелінійних нестаціонарних процесів та побудови їх математичних моделей

Відомо, що більшість сучасних процесів в економіці, фінансах та багатьох інших напрямах розвитку сучасного бізнесу відносяться до нестаціонарних. Щоб глибше зрозуміти, які саме процеси є нестаціонарними, потрібно також знати, які процеси називаються стаціонарними, і що означає сам термін «стаціонарність».

Стаціонарність визначається як якість процесу, в якому статистичні параметри (математичне сподівання і стандартне відхилення) не змінюються з часом [6]. Найважливішою властивістю стаціонарного процесу є те, що автокореляційна функція (ACF) залежить тільки від лагу і не змінюється з часом. Автокореляціна функція, або, як її ще називають, автокореляція (1.1) – це математичне подання ступеня подібності між даним часовим рядом і відсталою версією самого себе, зміщеною на певну величину незалежної змінної. Це те ж саме, що і розрахунок кореляції між двома різними часовими рядами, за винятком того, що автокореляція використовує один і той же часовий ряд двічі: один раз у початковій формі і один раз змістивши на один або більше періодів часу.

$$R(t_0) = \overline{f(t)f(t+t_0)} = \frac{1}{\tau}\int_0^{\tau} f(t)f(t+t_0)dt \qquad (1.1)$$

Також розрізняють слабко стаціонарні процеси – це такі, що мають постійне математичне сподівання і ACF (і, отже, дисперсію). По-справжньому стаціонарний (або сильно стаціонарний) процес має всі постійні моменти вищого порядку, включаючи дисперсію і математичне сподівання – таке визначення найчастіше можна знайти в літературі. Проте у літературі зазвичай не пояснюється, що сильно стаціонарні процеси ніколи не розглядаються на практиці і обговорюються тільки за їх математичними властивостями. Слабко стаціонарні процеси іноді спостерігаються в реальному світі і зазвичай вважаються «досить близькими» до стаціонарності в строгому сенсі (сильна стаціонарність). Крім того, стаціонарність є насправді відносним терміном, а не абсолютним, будь-який процес, який «дійсно» є стаціонарним, можна розглядати лише як стаціонарний, якщо вибіркові дані даного процесу дуже

великі порівняно з найменшою частотою в даних. Іншими словами, якщо збирати дані лише за короткий час, короткі порівняно з довжиною хвилі даних, то навіть стаціонарний процес виявиться нестаціонарним. Нарешті, не існує досліджень, які б обговорювали, які саме відхилення від стаціонарності, великі чи малі, можуть мати методи аналізу, які вимагають стаціонарності.

Для цілей аналізу властивість стаціонарності є дуже хорошою для даних, оскільки вона призводить до багатьох спрощуючих припущень. Знову ж таки, першим кроком у використанні будь-якої методології для аналізу часових рядів є перевірка того, чи є дані стаціонарними.

Зрозумівши, які процеси є стаціонарними, можна перейти до розгляду нестаціонарних процесів. Вони досить часто містять нелінійності стосовно змінних та параметрів. Нестаціонарність процесів, що аналізуються, проявляється у зміні їх статистичних параметрів на часових інтервалах дослідження. Так, зміна у часі математичного сподівання приводить до появи трендів різного рівня складності. Такі процеси називають ще інтегрованими. Якщо на інтервалі дослідження змінюється дисперсія процесу, то такі процеси називають гетероскедастичними. Очевидно, що процес може бути нестаціонарним одночасно стосовно двох і більше параметрів. Однак, у багатьох випадках існуючі методи прогнозування, які ґрунтуються на аналітичних процедурах моделювання, нечітких логічних правилах та раціональному експертному мисленні, не забезпечують отримання бажаного показника якості оцінок прогнозів. У зв’язку з цим виникає проблема підвищення якості оцінок прогнозованих значень. Для розв’язання задачі прогнозування на вищому рівні якості вимагає застосування сучасних методів і принципів системного аналізу до існуючих підходів та методів та коректного використання методів математичного моделювання процесів довільної природи на основі досягнень теорії адаптивного оцінювання і статистичного аналізу даних. Деякі можливості розв’язання задач адаптивного моделювання

і прогнозування розглядаються в роботах [7], [8], [9]. Однак, методи, подані в цих роботах, не грунтуються на системному підході до розв’язання задач аналізу даних і фактично не дають відповіді на основне запитання: як організувати процес обробки даних таким чином, щоб отримати кращі оцінки прогнозів в умовах наявності невизначеностей структурного, параметричного і статистичного характеру?

Згадані невизначеності зумовлені, як правило, нестаціонарністю досліджуваного процесу, пропусками даних, зашумленими даними, наявністю впливу випадкових зовнішніх збурень, екстремальними значеннями та відсутністю необхідних об’ємів вибірок даних. Ефективний метод адаптивного прогнозування за допомогою фільтра Калмана (ФК) представлено у роботі [10]. Для адаптації алгоритму оцінювання та прогнозування стану процесу використано обчислені в реальному часі оцінки статистичних характеристик збурень стану і шумів вимірів. Однак, застосування звичайного (не адаптивного) ФК має свої недоліки, наприклад, такі:

1. обчислення оцінок багатокрокових прогнозів вимагає використання проміжних оцінок;
2. у деяких випадках неможливо отримати прийнятні оцінки статистичних характеристик збурень стану і шумів вимірів;
3. існують проблеми обчислювального характеру при використанні нелінійних моделей у просторі станів.

### 1.3.1 Перевірка належності процесу до нестаціонарних

Зрозуміло, що маючи ті чи інші дані, спочатку необхідно встановити, чи є процес, який вони описують, стаціонарним чи нестаціонарним. Формальні тести гіпотез, як правило, концентруються на тестуванні одного виду альтернативи, але часто нечутливі до інших видів (але, звичайно, вони часто є дуже потужними для явищ, які вони призначені виявляти). Більш детально про ці тести описано в [11].

Як і в багатьох галузях статистики, можна дізнатись достатньо корисної інформації про дані, просто зобразивши їх на графіках. Наприклад, можна подивитися на графік часового ряду, щоб побачити, чи змінюється математичне сподівання або дисперсія часового ряду з часом. Іншим корисним показником є обчислення автокореляції (1.1) або спектра (або обох) на двох різних частинах часового ряду (які «здаються» стаціонарними). Якщо дві величини з різних регіонів виглядають дуже різними, то це дає деякі докази нестаціонарності. Додаткові графічні процедури можуть полягати в тому, щоб подивитися на певний часовий графік і визначити, чи має він постійність з плином часу, чи ні.

## 1.4 Існуючі комп’ютерні системи для побудови моделей фінансово-економічних процесів

Сьогодні важко знайти сферу людської діяльності, для якої не реалізовано відповідного програмного забезпечення. Все частіше будь яку обчислювальну роботу довіряють комп’ютеру, це і зрозуміло, адже досить важливими аспектами сучасного бізнесу є збереження часу, коштів, збільшення точності і продуктивності. Область моделювання нестаціонарних нелінійних процесів не є винятком.

Для моделювання рішень досить зручно використовувати програмний продукт GeNIe Modeler, який був розроблений в лабораторії систем прийняття рішень Піттсбургського університету, США і в 2015 році був ліцензований компанією BayesFusion. Перевагами додатку є інтуїтивний графічний інтерфейс, що включає в себе побудову дерев різних типів, діаграм впливу та байєсівських мереж.

GeNIe Modeler поставляється разом із SMILE(Structural Modeling, Inference, and Learning Engine) Engine – платформо-незалежною бібліотекою функцій, що реалізують графічні ймовірнісні та теоретичні моделі, такі як байєсівські мережі, діаграми впливу і моделі структурних рівнянь. Функції, визначені в SMILE API (Applications Programmer Interface), дозволяють створювати, редагувати, зберігати і завантажувати графічні моделі і використовувати їх для ймовірнісних міркувань і прийняття рішень в умовах невизначеності. Сама SMILE розроблена мовою C++, але існують обгортки цієї бібліотеки і на такі мови програмування, як: Java (jSMILE), .NET (SMILE.NET) і Python (PySMILE) [13].

### 1.4.1 Приклад побудови мережі Байєса з використанням програмного засобу GeNIe

Розглянемо, як користуватись комп’ютерною програмою GeNIe на прикладі покрокової побудови найпростішої байєсівської мережі. Для цього створимо мережу, що складається з наступних вершин: успішність студента на екзамені та передбачення отримування ним стипендії.

Для побудови мережі слід виконати такі кроки:

1. Потрібно відкрити програму, натиснути на кнопку створення вершини, після чого натиснути на будь-якій частині полотна, як зображено на Рисунку 1.1.

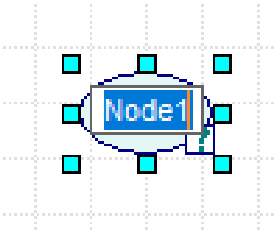



Рисунок 1.1 - Початковий вигляд вершини мережі.

2. Далі можна змінити назву вершини, розміри і встановити можливі стани, вказавши при цьому ймовірність настання кожного з них. Для прикладу ми взяли ймовірність успіху студента на екзамені 0,8 і, відповідно, ймовірність невдачі 0,2, що зображено на рисунку 1.2.

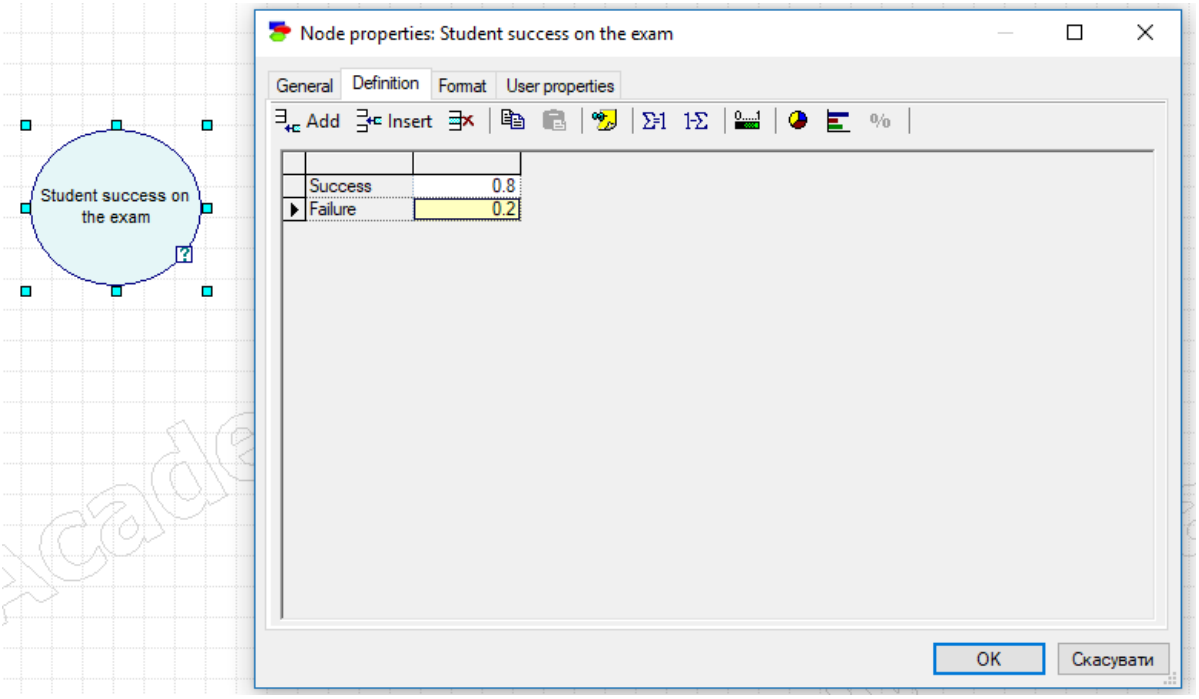



Рисунок 1.2 – Задавання ймовірнісних параметрів станів вершини.

Програма перевіряє коректність введених параметрів і попередить, якщо сума ймовірностей не буде дорівнювати 1. Серед корисних функцій цього вікна є клавіші «Σ = 1», яка перераховує пропорційно введені значення так, щоб сума була рівною 1 (наприклад, можна ввести значення у відсотках (80 і 20), які автоматично перетворяться у ймовірності), та «1 – Σ», яка шукає значення ймовірності, віднімаючи від одиниці суму значень інших параметрів (для прикладу можна було записати в першому рядку ймовірність 0,8, а в другому просто натиснути цю клавішу і отримати значення 1-0,8 = 0,2).

3. Перша вершина готова, можна переходити до наступної. Аналогічно кроку 1 створюємо нову вершину і, вибравши стрілку на панелі інструментів, з'єднуємо вершини між собою, отримаємо мережу, зображену на рисунку 1.3.

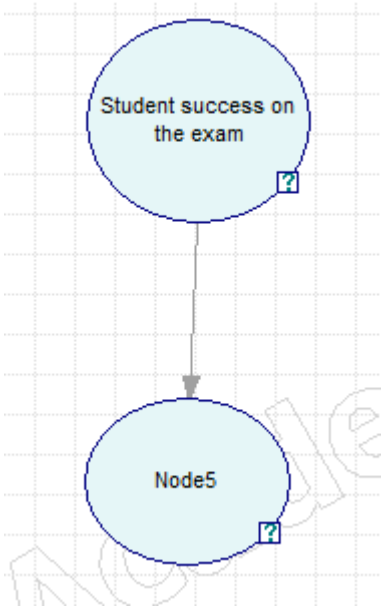



Рисунок 1.3 – Дві вершини мережі Байєса, з'єднані між собою.

4. Далі потрібно заповнити ймовірнісні параметри другої вершини. Оскільки стрілка, що зв'язує вершини, вказує на другу вершину, то вона

залежить від першої і ймовірності її станів можуть відрізнятись залежно від стану першої вершини, тому в таблиці ймовірностей рядки відповідають станам другої вершини вершини («High chances», «Average chances», «Low chances»), а стовпчики – станам першої(«Success», «Failure»), як показано на рисунку 1.4.

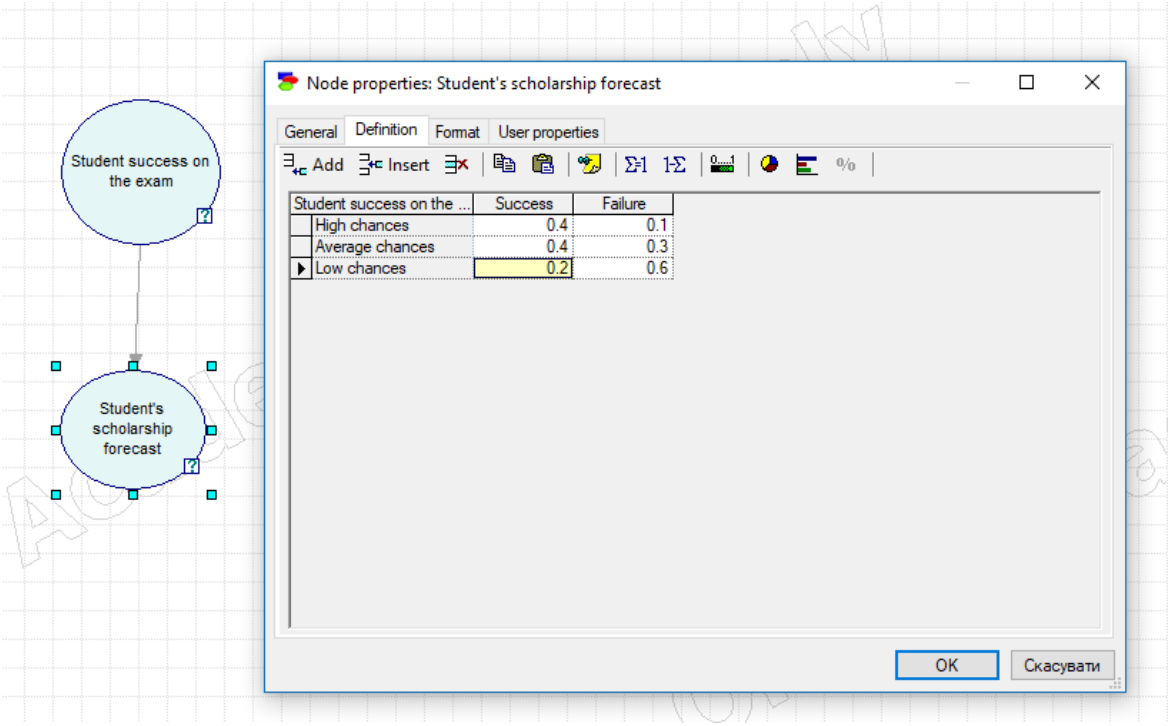



Рисунок 1.4 – Задавання ймовірнісних параметрів станів залежної вершини.

5. Тепер знайдемо відповідь на питання: «Яка ймовірність успішності студента повинна бути на екзамені, щоб отримати високі шанси вийти на стипендію?». Для цього необхідно правою клавішею мишки натиснути на вершину «Student’s scholarship forecast», обрати в контекстному меню «Set Evidence» і натиснути «High_chances», як це зроблено на рисунку 1.5.

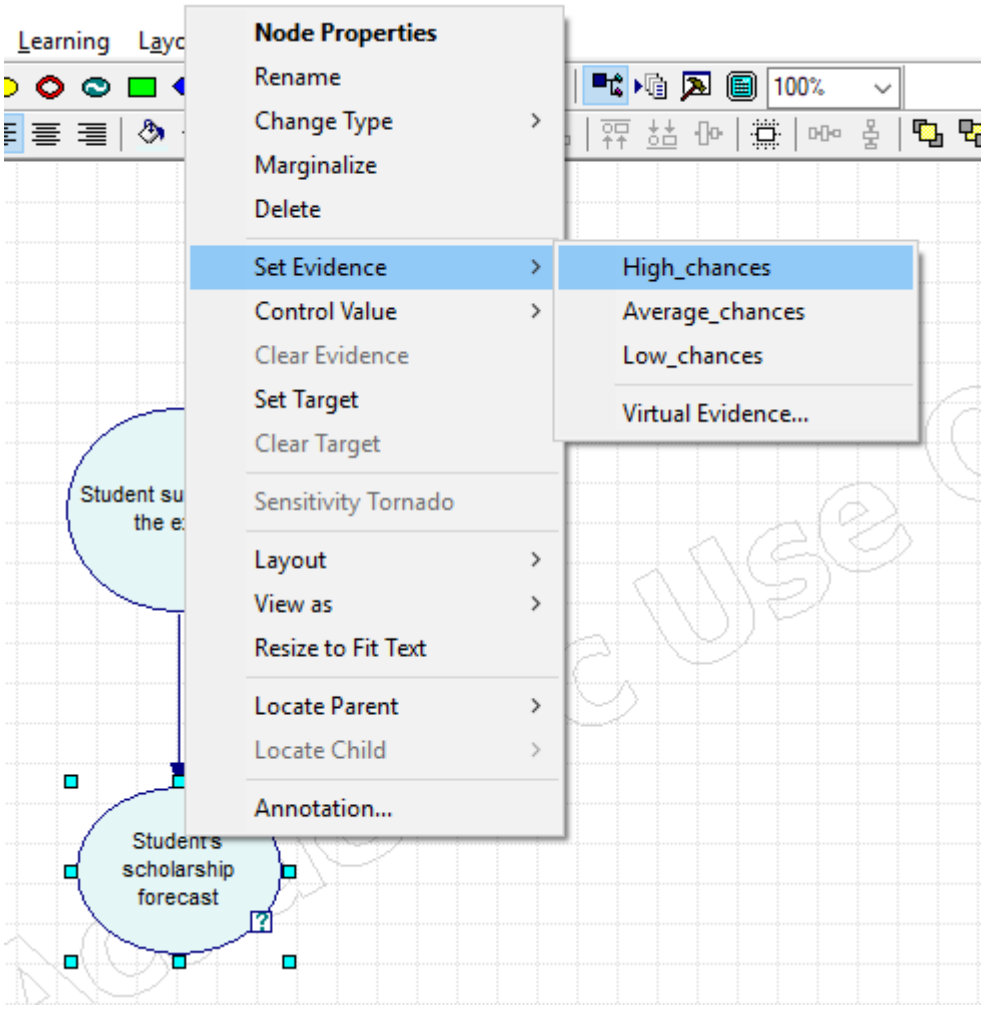



Рисунок 1.5 – Встановлення значення вершини, для якої потрібно знайти прогноз.

У правому нижньому кутку вершини «Student success on the exam» знак питання зміниться на знак «галочка». При наведені курсору на цей знак побачимо ймовірність успішності студента для високих шансів отримання стипендії, що зображено на рисунку 1.6.

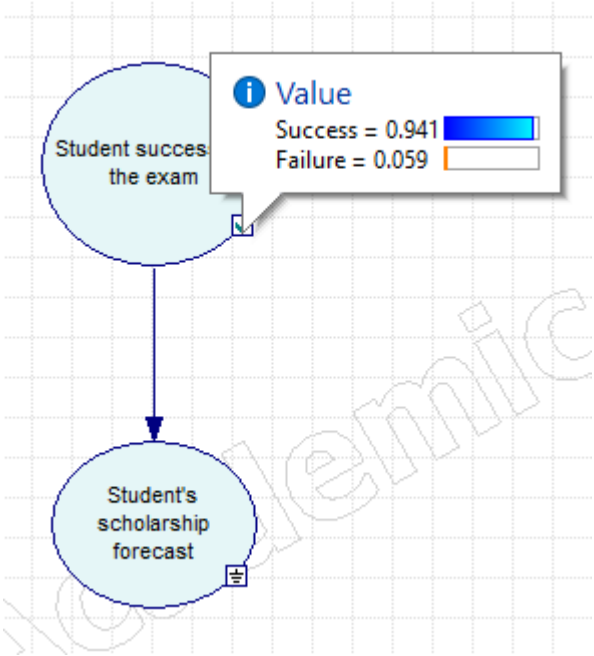



Рисунок 1.6 – Результуючий розподіл ймовірностей.

Також можна переглянути ймовірності у вигляді кругової діаграми, перейшовши у властивості вершини, вкладка «Value», як це зроблено на рисунку 1.7.

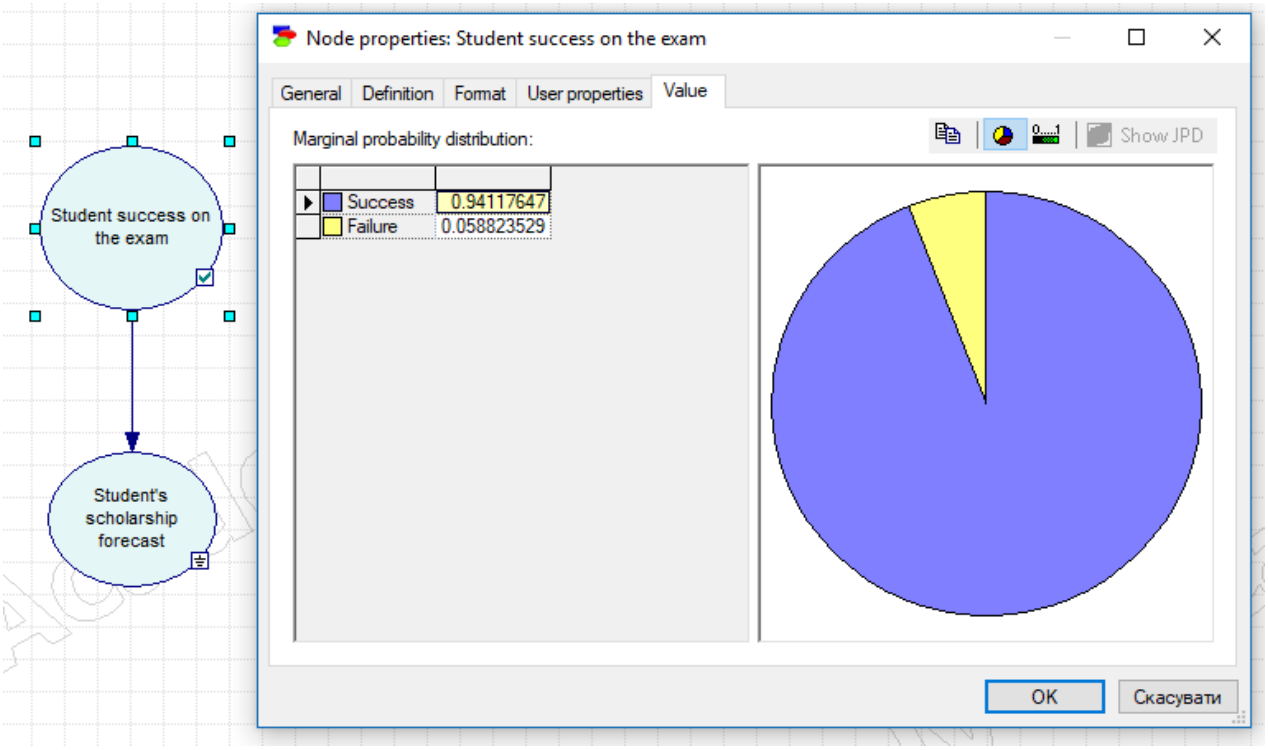



Рисунок 1.7 – Перегляд ймовірностей у вигляді кругової діаграми.

6. Можна змінити вигляд вершин на більш наглядний, який показаний на рисунку 1.8.

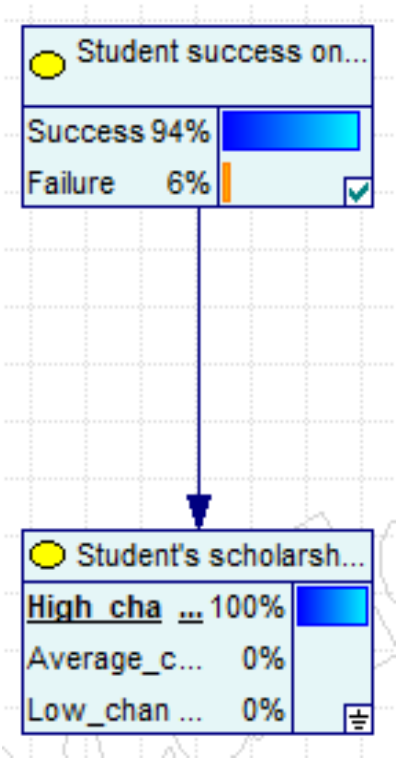



Рисунок 1.8 – Відображення вершин із ймовірностями.

Для цього у меню «Node» обрати «View as»/«Bar Chart», як показано на рисунку 1.9.

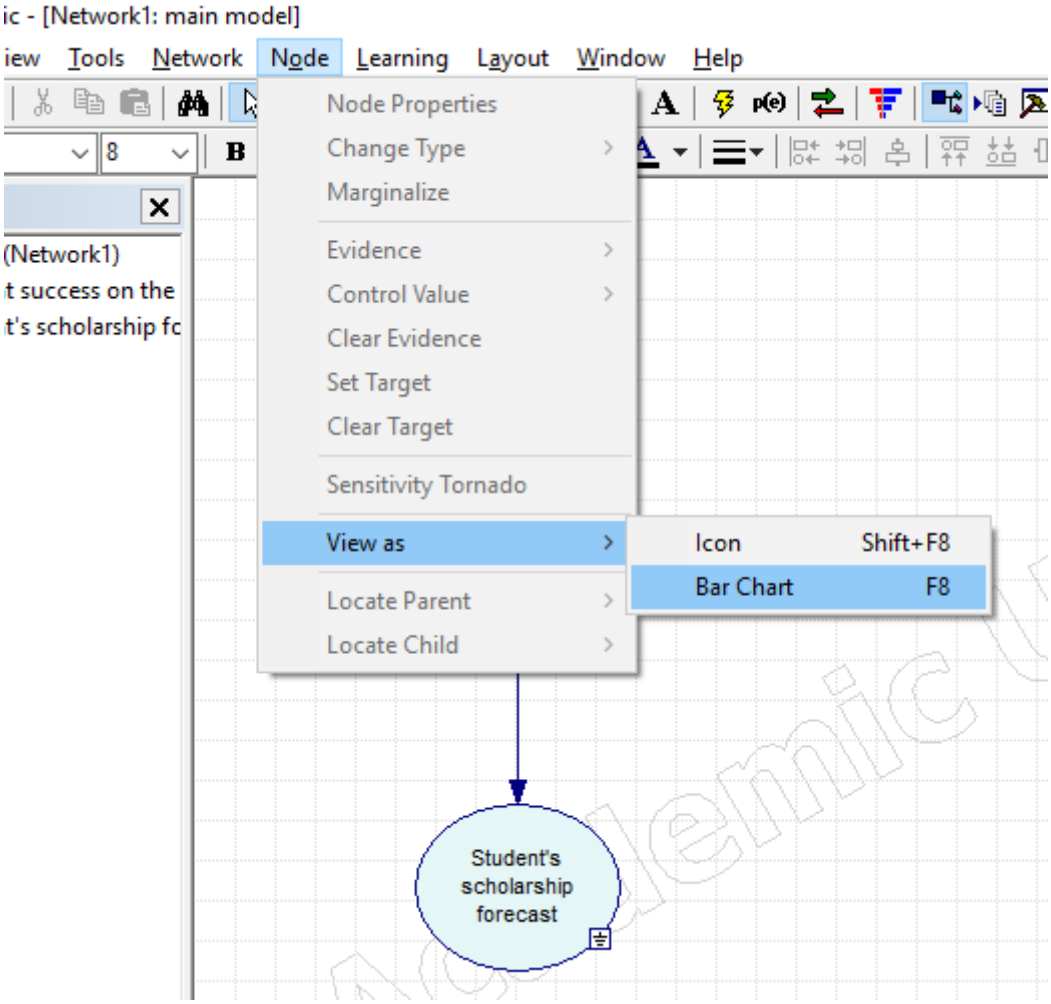



Рисунок 1.9 – Змінення відображення вершин.

При цьому стани вершин будуть відображатись на самих вершинах і ми зможемо, натиснувши на якийсь із них, вибрати стан, який ми обираємо як доказ, він стане підкресленими і його ймовірність 100%, і ймовірності інших параметрів перерахуються, що продемонстровано на рисунку 1.10.

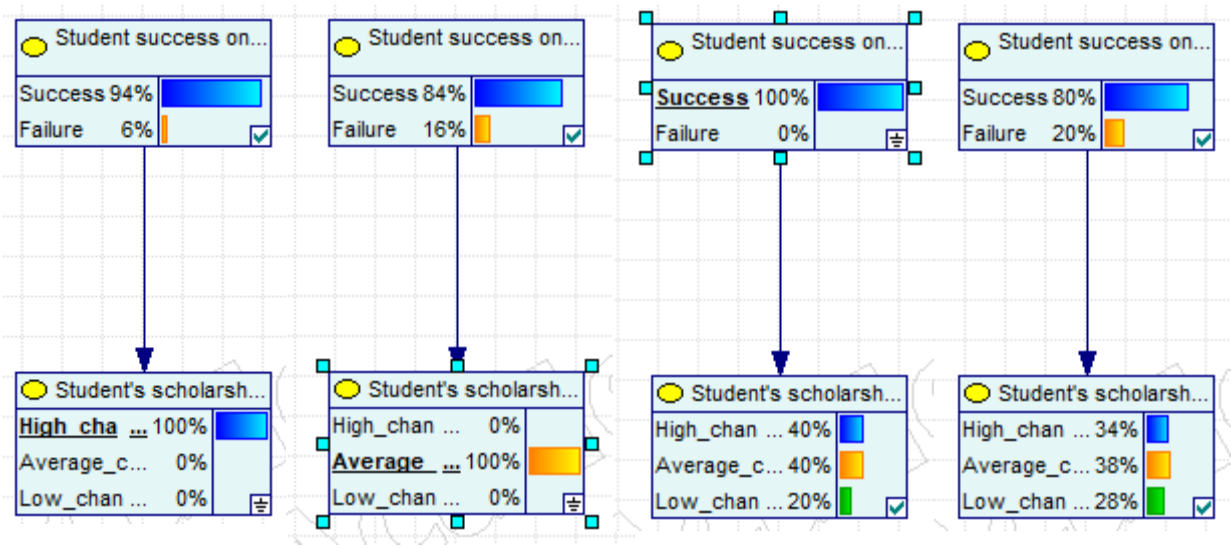



Рисунок 1.10 – Різні варіанти розподілу ймовірностей в залежності від обраного основного стану.

## 1.5 Постановка задачі дослідження

Для визначеності кредитоспроможності клієнта необхідно проаналізувати ряд факторів, таких як:

- статус поточного існуючого рахунку;
- тривалість кредиту у місяцях;
- кредитна історія;
- призначення кредиту;
- сума кредиту;
- ощадний рахунок/облігації;
- роки працевлаштування;
- ставка погашення у відсотках від наявного доходу;
- особистий статус і стать;

- інші сторони (поручителі);
- власність;
- вік;
- інші плани розстрочок;
- наявність житла;
- кількість існуючих кредитів у цьому банку;
- працевлаштування;
- кількість людей, що підлягають обслуговуванню;
- телефон;
- працевлаштування за кордоном;

Для визначення ймовірності повернення кредиту необхідно мати повний набір перерахованих вище даних та відповідний програмний продукт.

Необхідно розробити евристичний метод побудови байєсовської мережі, що включає в себе два етапи. На першому виконується обчислення значення спільної інформації для всіх вершин, після чого виконується цілеспрямований пошук, що використовує оціночну функцію у вигляді оцінки мінімальної довжини.

Потім ставиться задача розробки методу ймовірнісного висновку в байєсівській мережі. Спочатку виконується обчислення матриці емпіричних значень спільного розподілу ймовірностей по всій мережі. Згодом здійснюється обчислення значень ймовірностей всіх можливих станів вершин.

Таким чином, отримали наступну задачу: на основі аналізу даних кредитних історій 1000 клієнтів німецьких банків необхідно побудувати байєсівську мережу для прогнозування повернення кредиту новим клієнтом, оцінити похибку та порівняти точність прогнозу з іншими методами інтелектуального аналізу даних.

## 1.6 Висновки до розділу 1

В сучасному світі всі процеси можуть бути описані математично. Фінансова сфера людської діяльності завжди потребує досить точних обчислень та прогнозів. Для того, щоб можна було вести подальший аналіз процесу та мати змогу спрогнозувати його результати, потрібно відтворити процес у вигляді математичної моделі, яка повністю описує його поведінку. Математичні моделі, як і процеси, бувають різного характеру: лінійні, нелінійні, стаціонарні, нестаціонарні. Звісно в навколишньому середовищі більш розповсюджені нелінійні нестаціонарні процеси. Тому розглянутий в даній роботі матеріал стосується моделювання та прогнозування нелінійних нестаціонарних процесів.

У даному розділі також було розглянуто основні підходи обробки інформації, методи та етапи інтелектуального аналізу даних та особливості нелінійних нестаціонарних процесів та їх моделей. Також було відображено покроковий процес побудови моделі у вигляді байєсівської мережі та прогнозування з її використанням у прикладного програмному засобі GeNIe.

# РОЗДІЛ 2 ВИБІР І ОПИС МАТЕМАТИЧНИХ МОДЕЛЕЙ ДЛЯ ВИБРАНИХ ПРОЦЕСІВ

## 2.1 Найпростіші математичні моделі для нелінійних нестаціонарних процесів

### 2.1.1 Модель авторегресії (AR – Autoregressive model)

Першою моделлю, яку ми розглянемо, є модель авторегресії (AR). Вона є однією з найбільш інтуїтивно зрозумілих моделей. Основна ідея полягає в тому, що ми будемо моделювати значення в момент часу t як лінійну функцію його попередніх значень і деякого випадкового шуму. Розглянута модель AR(N) визначається наступною формулою:

$$y(k) = a_0 + \sum_{i=1}^{N} a_i y(k-i) + b_1 k + \cdots + b_m k^m + \varepsilon(k);$$

У цій формулі $a_i$ - коефіцієнти авторегресії, $y(k)$ - досліджуваний ряд, $k = 1,\ 2,\ 3,\ \ldots$ - дискретний час, $t = kT_s$, де $T_s$ – час виміру спроби, а N - порядок (довжина) фільтра, який, як правило, набагато менший довжини серії. Шумовий член або залишок, епсилон у вищезазначеній формулі, майже завжди вважається гаусівським білим шумом. Як бачимо, у цій моделі присутня стаціонарність, оскільки існує тренд порядку $m$ (степінь $k$).

Існує ряд методів для обчислення коефіцієнтів AR. Найбільш поширеним є метод найменших квадратів, що базується на рівняннях Юла-Уокера, які можна записати в матричній формі, як:

$$\begin{pmatrix} 1 & r_1 & r_2 & r_3 & \dots & r_{N-1} \\ r_1 & 1 & r_1 & r_2 & \dots & r_{N-2} \\ r_2 & r_1 & 1 & r_1 & \dots & r_{N-3} \\ r_3 & r_2 & r_1 & 1 & \ddots & r_{N-4} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{N-1} & r_{N-2} & r_{N-3} & r_{N-4} & \dots & 1 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ \vdots \\ a_N \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ \vdots \\ r_N \end{pmatrix}$$

### 2.1.2 Модель ковзного середнього (MA – Moving Average)

У статистиці метод ковзного середнього широко застосовується для «згладжування» функції шляхом створення серії середніх для різних підмножин повного набору даних. Перший елемент ковзного середнього отримують, взявши середнє значення початкового фіксованого набору елементів підмножини ряду чисел. Тоді підмножина модифікується шляхом «переміщення вперед», тобто виключаючи перше число серії і включаючи наступне значення в підмножині. Модель MA(q) визначається наступною формулою:

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

Ковзне середнє зазвичай використовується з даними часових рядів для згладжування короткострокових коливань і виділення довгострокових тенденцій або циклів. Наприклад, дана модель часто використовується в технічному аналізі фінансових даних, таких як ціни на акції, прибутки або обсяги товарообротів. Вона також використовується в економіці для визначення валового внутрішнього продукту, зайнятості працівників або

інших макроекономічних часових рядів. Математично, ковзне середнє значення є типом згортки і тому його можна розглядати як приклад фільтра низьких частот, який використовується для обробки сигналів.

Існує декілька різновидів ковзного середнього:

– Просте ковзне середнє (SMA – Simple Moving Average)

У цій моделі значення шукається як незважене середнє попередніх n показників. Для набору $a_M, a_{M-1}, a_{M-2}, a_{M-3}, \dots, a_{M-(n-1)}$ має місце наступна формула розрахунку ковзного середнього:

$$\overline{a_M} = \frac{a_M + a_{M-1} + a_{M-2} + a_{M-3} + \dots + a_{M-(n-1)}}{n}$$

– Сукупне ковзне середнє (CMA - Cumulative Moving Average)

У сукупному ковзному середньому подана інформація утворює впорядкований потік даних, і користувач хоче отримати середнє значення всіх даних до поточної вхідної точки. Наприклад, інвестор хоче, щоб середня ціна всіх біржових операцій була з точністю до поточного часу. Формула розрахунку для n даних:

$$CMA_n = \frac{x_1 + \dots + x_n}{n}$$

Для розрахунку $n+1$ значення можна використати попередньо розраховане значення:

$$CMA_{n+1} = \frac{x_{n+1} + n \cdot CMA_n}{n+1}$$

– Зважене ковзне середнє (WMA - Weighted Moving Average)

Зважене ковзне середнє розраховується подібно до простого ковзного середнього, за винятком того, що даним, розташованим у різних місцях ковзного вікна, надаються певні вагові коефіцієнти. У аналізі фінансових даних часто як ваги використовують спадну арифметичну прогресію:

$$WMA_M = \frac{na_M + (n-1)a_{M-1} + \cdots + 2a_{M-n+2} + a_{M-n+1}}{n + (n-1) + \cdots + 2 + 1}$$

2.1.3 Модель авторегресії з ковзним середнім (ARMA - Autoregressive Moving Average)

Авторегресія з ковзним середнім (ARMA) - це модель прогнозування, в якій до даних одночасно застосовуються обидва попередні методи авторегресії (AR) і ковзного середнього (MA). Наприклад, ARMA(p, q) складається із p складових авторегресії і q – ковзного середнього. Ця модель містить AR(p) та MA(q):

$$y(k) = a_0 + \sum_{i=1}^{p} a_i y(k-i) + \sum_{i=1}^{q} \theta_i \varepsilon(k-i) + b_1 k + \cdots + b_m k^m + \varepsilon(k);$$

Модель ARMA є інструментом для розуміння і, можливо, прогнозування майбутніх значень у цій серії. Частина AR включає регресування змінної на власних попередніх значеннях. Інша частина, тобто MA, передбачає моделювання помилки, як лінійної комбінації помилок, що відбуваються одночасно і в різні часи в минулому. Як бачимо, у цій моделі також явно видно присутність стаціонарність, оскільки існує тренд порядку $m$ (степінь $k$).

### 2.1.4 Логістична регресія

Логістична регресія є потужним інстументом для моделювання і прогнозування нелінійних нестаціонарних процесів будь-якої складності. За основу моделі взято функцію логістичного розподілу:

$$\varphi\big(x(k,z)\big) = \frac{1}{1+e^{-x(k,z)}}$$

Математично логістична регресія записується наступним чином:

$$x(k) = \alpha_0 + \alpha_1 z_1(k) + \cdots + \alpha_m z_m(k) + \varepsilon(k).$$

Особливостю методу є те, що його, як і байєсівські мережі, можна використовувати для категоріальних наборів даних, у яких залежна змінна представлена у вигляді скінченної множини значень.

## 2.2 Байєсівські мережі для моделювання нелінійних нестаціонарних процесів

Для вирішення поставленої задачі було обрано один із найбільш оптимальних і наглядних алгоритмів – мережу Байєса.

Байєсівські мережі – перспективний ймовірнісний інструментарій для моделювання складних ієрархічних процесів (статичних і динамічних) з невизначеностями довільного характеру. Мережа будується у вигляді

спрямованого ациклічного графа, який відображає причинні зв'язки між вузлами (змінними) процесу, що досліджується.

Термін "байєсівська мережа" (Bayesian Network) був запропонований американським вченим Джуді Перлом у 1985 році з метою узагальнення трьох аспектів:

- об'єктивної природи вхідних даних;
- отримання достовірної інформації при застосуванні теореми Байєса;
- ідеї застосування аналізу причин та наслідків, запропонованої в 1763 році у посмертній роботі англійського священика Томаса Байєса.

Байєсівські мережі з'явилися на стику двох наук:

- теорії ймовірностей;
- теорії графів (розділ дискретної математики) [1].

Використання байєсівських мереж як інструменту інтелектуального аналізу даних передбачає розв'язання двох математичних задач, таких як: побудова структури БМ та формування ймовірнісного висновку.

Перша задача є задачею нелінійної поліноміальної складності. За допомогою рекурентної формули Робінсона можна обрахувати кількість всіх можливих нециклічних моделей, які потрібно проаналізувати. Проте на практиці через обмеження обчислювальних ресурсів повний перебір моделей можна здійснити лише для мережі з кількістю вузлів не більш ніж з 7.

Задача формулювання ймовірнісного висновку є складною і відноситься до задач прийняття рішень. Існуючі методи побудови байєсівської мережі і формулювання висновку вимагають трудомістких обчислень, тому актуальними є пошук і розробка методів, що дозволяють зменшити обчислювальну складність при моделюванні процесів різної природи мережами Байєса.

Формально Байєсівська мережа визначається як пара < *G*, *B* >, в якій перша складова *G* є ациклічним графом, який відповідає змінним. Друга – *B* є множиною параметрів, що визначають мережу. Компонентами *B* є параметри $\Theta_{x^i|parent(X^i)} = P\left(x^i|parent(X^i)\right)$ для кожного можливого значення $x^i$ із $X^i$ і $parent(X^i)$ із множини батьківських вершин $X^i$ в графі *G*. Повна спільна ймовірність БМ визначається за формулою [5]:

$$P_B(X^1, \dots, X^N) = \prod_{i=1}^{n} P_B\left(X^i|Parent\left(X^i\right)\right)$$

2.2.1 Теорема Байєса як інструмент формування ймовірнісного висновку БМ

Для обрахунку ймовірності одночасної появи двох незалежних подій *A* і *B* використовують наступну формулу:

$$p(A, B) = p(A) \cdot p(B)$$

У випадку залежності подій формула дещо змінюється, оскільки настання однієї з подій впливає на іншу:

$$p(A, B) = p(A) \cdot p(B|A)$$

Враховуючи комутативність операції, попередню формулу можна продовжити:

$$p(A,B) = p(A) \cdot p(B|A) = p(B) \cdot p(A|B) \qquad (2.1)$$

З формули (2.1) випливає формула теореми Байєса для двох подій:

$$p(B|A) = \frac{p(B) \cdot p(A|B)}{p(A)}$$

У літературі зустрічається наступне визначення теореми Байєса:

Нехай випадкова подія А може відбуватися тільки разом із однією з гіпотез $H_1$, $H_2$, ..., $H_n$ , які утворюють повну групу подій деякого стохастичного експерименту. Нехай відомі апріорні, тобто ще до проведення експерименту, ймовірності гіпотез $P(H_i)$. В результаті проведеного експерименту відбулася подія А. Її поява впливає на ймовірності гіпотез. Постає питання: як змінюються ймовірності гіпотез після того, як відбулася подія А і, отже, відбулась одна з гіпотез $H_i$, i=1, 2, …, n? Обчислення апостеріорних, тобто після дослідних, імовірностей гіпотез $P(H_i / A)$ здійснюється за формулою Байєса [12]

$$P(H_i|A) = \frac{P(H_i)P(A|H_i)}{\sum_{k=1}^{n} P(H_k)P(A|H_k)}, \quad i = 1,2, \dots, n.$$

Вищезгадана теорема активно використовується при формуванні ймовірнісного висновку на основі мережі Байєса.

### 2.2.2 Приклад використання мережі Байєса

Розглянемо приклад побудови мережі Байєса. Припустимо, що існує дві події, які впливають на результат футбольного матчу: стан покриття футбольного поля та опади у вигляді дощу. Кожна з подій має свою ймовірність появи і може залежати від інших подій. Позначимо події наступним чином:

- R – Дощ (від англійського слова Rain);
- Q – Стан покриття футбольного поля (від англійського слова Quality);
- W – Перемога в матчі (від англійського слова Win).

Зобразимо граф мережі на рисунку 2.1, у якому вершини є нашими подіями, а ребра показують залежність між подіями.

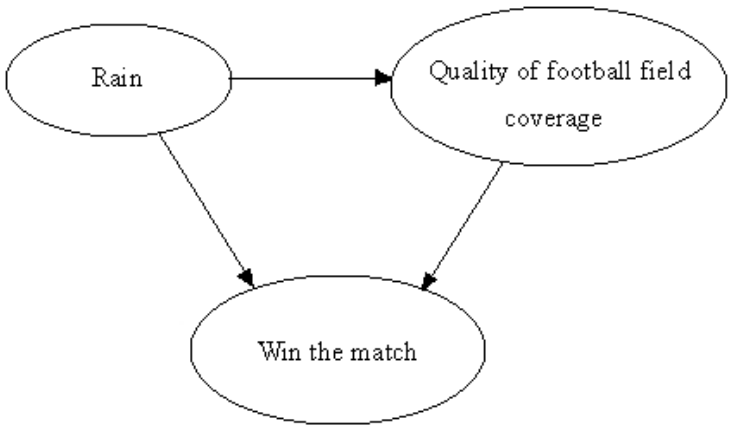



Рисунок 2.1 – Мережа Байєса для 3 подій

Для кожної події є своя ймовірнісна таблиця 2.1 – 2.3:

Таблиця 2.1 – Ймовірності для події Rain

| T | F |
|---|---|
| 0.1 | 0.9 |

Таблиця 2.2 – Ймовірності для події Quality of football field coverage

| Rain | Good | Bad |
|---|---|---|
| T | 0.2 | 0.8 |
| F | 0.9 | 0.1 |

Таблиця 2.3 – Ймовірності для події Win the match

| Rain | Quality of football field coverage | T | F |
|---|---|---|---|
| F | Bad | 0.4 | 0.6 |
| F | Good | 0.9 | 0.1 |
| T | Bad | 0.05 | 0.95 |
| T | Good | 0.2 | 0.8 |

Знаючи ймовірність настання кожної з подій, можна дати відповіді на питання:

– Якою є ймовірність того, що пішов дощ, якщо гра виграна?

$$P(R = T|W = T) = \frac{P(W = T, R = T)}{P(W = T)} = \frac{\sum_{Q \in \{Good,Bad\}} P(W = T, Q, R = T)}{\sum_{\substack{Q \in \{Good,Bad\} \\ R \in \{T,F\}}} P(W = T, Q, R)}$$

Підставивши у формулу відповідні табличні значення, матимемо:

$$P(R = T|W = T)$$

$$= \frac{0.2 \cdot 0.2 \cdot 0.1 + 0.05 \cdot 0.8 \cdot 0.1}{0.2 \cdot 0.2 \cdot 0.1 + 0.05 \cdot 0.8 \cdot 0.1 + 0.9 \cdot 0.9 \cdot 0.9 + 0.4 \cdot 0.1 \cdot 0.9} =$$

$$= \frac{0.004 + 0.004}{0.004 + 0.004 + 0.729 + 0.036} = \frac{0.008}{0.773} \approx 1\%.$$

## 2.3 Алгоритми побудови байєсівських мереж

Методи побудови структури мереж Байєса поділяються на дві підгрупи:

- методи, що беруть за основу оціночні функції (search & scoring);
- методи, що застосовують тест на умовну незалежність (dependency analysis).

### 2.3.1 Методи на основі оціночних функцій (search & scoring)

Чу і Ліу (Chow and Liu) в 1968 році запропонували алгоритм для побудови БМ у вигляді дерева, заснований на використанні значень взаємної інформації між вершинами. В якості рішення метод видає структуру зі значенням спільного розподілу ймовірності мережі, найбільш відповідного навчальним даним Побудова структури БМ здійснюється за кроків, де – кількість вершин мережі, однак цей алгоритм не працює для багатозв’язаних БМ.

В 1988 році Рібан і Перл (Rebane and Pearl) запропонували вдосконалений змінений алгоритм Чу і Ліу для побудови БМ у вигляді полі-дерева.

Купер і Гершкович (Cooper and Herskovits) в 1990 році розробили алгоритм Кутато (Kutato). На етапі ініціалізації алгоритму вважається, що всі вершини БМ незалежні, після чого обчислюється ентропія цієї мережі. Потім виконується додавання дуг між вершинами в мережі таким чином, щоб мінімізувати ентропію БМ. Для роботи алгоритму потрібна наявність ВМВ.

Купер і Гершкович в 1992 році запропонували широко відомий алгоритм К2, який виконує пошук структури з максимальним значення функції Купера-Гершковича (КГ). Для роботи алгоритма потрібна наявність ВМВ.

В 1994 році був запропонований алгоритм HGC. Цей алгоритм суттєво відрізняється від інших алгоритмів застосовуючи оціночні функції, тим що вперше саме в ньому були використані два нових поняття параметричної модульності (parametric modularity) та рівнозначності подій (event equivalence). Інші дослідники досить довго не використовували одночасно цих понять. Але одночасне застосування цих понять дозволяє об’єднувати статистичну інформацію та експертних знань для побудові БМ.

Вонг і Сианг (Wong and Xiang) запропонували в 1994 році алгоритм для побудови Марковських мереж з використанням значення ентропії та I-map. Граф імовірнісної моделі називається незалежною картою (independency map, скорочено I-map) коли з незалежності вершин графа витікає незалежність земних моделі. Цей алгоритм дозволяє процес який моделюється представити у вигляді I-map і у випадку, коли мережа являється однозв’язною гарантовано будується БМ. Разом із Чу (Chu) Сіанг розробили в 1997 році більш швидкий варіант запропонованого алгоритму.

Алгоритм Лема-Бахуса (Lam-Bacchus), запропонований в 1996 році, виконує евристичну побудову структури мережі, використовуючи значення

взаємної інформації між вершинами, а в якості функції оцінки використається функція опису мінімальною довжиною (minimum description length).

Алгоритм Бенедикта (Benedict), запропонований в 1996 році, виконує евристичний пошук на основі ВМВ, аналізуючи умовні незалежності в структурі мережі на основі d-розділення, а в якості функції оцінки використається ентропія.

СВ алгоритм запропонований в 1995 році. Він використає ТУН між вершинами мережі, для побудови ВМВ. Для побудови структури мережі використається функція КГ.

Алгоритм Фрідмана-Голдшмідта (Friedman-Goldszmidt) запропонований в 1996 році. Для побудови мережі використається аналіз її локальних підструктур, а в якості функції оцінки використається функція опису мінімальною довжиною (ОМД) та оцінка Байєса.

В алгоритмі WKD, запропонованому в 1996 році, в якості функції оцінки при побудові мережі використається функція повідомлення мінімальної довжини (minimum message length), яка схожа на ОМД.

Алгоритм Сузукі (Suzuki), запропонований в 1999 році, заснований на методі гілок та границь (branch and bound method) для завдання послідовності побудови структури мережі, а в якості функції оцінки використається ОМД.

Також існує ціла низка різноманітних поглинаючих алгоритмів (greedy algorithm) в яких для оцінювання можуть використовуватися різноманітні функції, наприклад максимальної правдоподібності або байєсівський інформаційний критерій.

### 2.3.2. Методи з використанням тестів на умовну незалежність (dependency analysis )

В 1983 році Вермут і Лоуренс (Wermuth and Lauritzen) запропонували алгоритм для побудови структури БМ застосовуючи ТУН. Цей алгоритм виконує послідовний перебір ВМВ. Для кожної пари вершин та , таких що (тобто це предок відносно ), виконується обчислення значення умовної незалежності. Цей алгоритм гарантує побудову БМ по навчальним даним, але при цьому буде потрібно виконати велику кількість ТУН між вершинами, що можливо лише у випадку, коли мережа складається з невеликої кількості вершин.

В 1988 році Перл (Pearl) запропонував алгоритм побудови скінченого спрямованого ациклічного графа (boundary DAG algorithm). Цей алгоритм будує БМ маючи ВМВ та функцію спільного розподілу (або достатньо велику навчальну вибірку даних). В купі з любим не досить складним методом пошуку цей алгоритм позбавляється проблеми в необхідності розрахунку великої кількості ТУН застосовуючи алгоритм Вермута і Лоуренса. Однак проблема в необхідності обчислення великої кількості ТУН з'являється при використанні цього алгоритму для побудови Марковських мереж, тобто мереж зі скритими вузлами (hidden node).

В 1990 році був запропонований SRA алгоритм, який являється модифікацією алгоритму скінченого спрямованого ациклічного графа. Цей алгоритм має менш жорсткі вимоги до ВМВ, для побудови БМ достатньо мати частково впорядковану множину вершин та ще деякі обмеження. Побудова БМ виконується послідовним додаванням дуг між вершинами, для цього використовується евристичний пошук. Але алгоритм виконує експоненціальну кількість розрахунків ТУН.

Алгоритм "Конструктор" (constructor algorithm) запропонований в 1990 році, дуже схожий на алгоритм побудови скінченого спрямованого ациклічного графа. Він намагається замість БМ побудувати Марковську

мережу. Відмінність цього метода від інших методів які використовують ТУН в тому що він не виконую надлишкові ТУН та йому не потрібна ВМВ.

Алгоритму SGS, запропонованому в 1990 році, для побудови структури не потрібна наявність ВМВ, але замість неї йому доводиться виконувати експоненціальну кількість тестів на умовну незалежність між вершинами.

PC алгоритм розроблений в 1991 році представляє собою вдосконалений варіант SGS алгоритму. Цей алгоритм спеціально розроблявся для побудови розріджених БМ (sparse BN), тобто для мереж з невеликою кількістю дуг між вершинами [5].

## 2.4 Критерії якості рішення задачі

Для оцінки якості роботи будь-якого методу побудови імовірнісного висновку можуть використовуватися такі величини як середньо квадратична похибка, KL-відстань або квадратична відстань Хелліндже́ра (Hellinger) [Bidyuk_2006].

Середньо квадратична похибка $MSE$:

$$MSE = \frac{1}{\sum_{X^{(i)} \in X \setminus E} card(A^{(i)})} \cdot \sum_{X^{(i)} \in X \setminus E} \left( \sum_{\forall x^{(i)} \in X^{(i)}} \left( P\left(x^{(i)} | e\right) - \hat{P}\left(x^{(i)} | e\right) \right)^2 \right).$$

*KL - відстань* $D_K$ між значенням ймовірності $P(X^{(i)}|e)$ та оцінкою $\hat{P}(X^{(i)}|e)$:

$$D_K\left(P\left(X^{(i)}|e\right),\hat{P}\left(X^{(i)}\middle|e\right)\right)=\sum_{\forall x^{(i)}\in A^{(i)}}\left(P\left(x^{(i)}|e\right)\cdot\log\left(\frac{P\left(x^{(i)}|e\right)}{\hat{P}\left(x^{(i)}\middle|e\right)}\right)\right).$$

*KL - відстань* $D_K$ всієї БМ:

$$D_K\left(P,\hat{P}\right)=\frac{1}{card(X\backslash E)}\cdot\sum_{X^{(i)}\in X\backslash E}D_K\left(P\left(X^{(i)}|e\right),\hat{P}\left(X^{(i)}\middle|e\right)\right).$$

*Квадратична відстань Хелліджера* $D_H$ між значенням ймовірності $P(X^{(i)}|e)$ та оцінкою $\hat{P}(X^{(i)}|e)$:

$$D_H\left(P\left(X^{(i)}|e\right),\hat{P}\left(X^{(i)}\middle|e\right)\right)=\sum_{\forall x^{(i)}\in A^{(i)}}\left(\sqrt{P(x^{(i)}|e)}-\sqrt{\hat{P}(x^{(i)}|e)}\right)^2.$$

Квадратична відстань Хелліджера $D_H$ всієї БМ:

$$D_H\left(P,\hat{P}\right)=\frac{1}{card(X\backslash E)}\cdot\sum_{X^{(i)}\in X\backslash E}D_H\left(P\left(X^{(i)}|e\right),\hat{P}\left(X^{(i)}\middle|e\right)\right).$$

В наведених формулах $X=\{X^{(1)},...,X^{(N)}\}-$ множина всіх вершин графа БМ *G*, де кожна $j$-та вершина мережі ( $j=1,...,N$ ) має $A^{(j)}=\{0,1,...,\alpha^{(j)}-1\}$ ( $\alpha^{(j)}\geq 2$) станів, запис $card(A^{(j)})$ означає потужність множини $A^{(j)}$ (кількість елементів з яких складається множина), $E\subset X, E=e-$ множина подій (інстанційовані вершини), $P(X^{(i)}|e)-$ значення ймовірності вершини $X^{(j)}$ за умови відбуття події $E=e$, $\hat{P}(X^{(i)}|e)-$ значення оцінки ймовірності.

## 2.5 Висновки до розділу 2

Даний розділ присвячений опису основних математичних моделей, що можуть бути застосовані до нелінійних нестаціонарних процесів, а саме: авторегресія, авторегресія з ковзним середнім та логістична регресія. Більш детальніше розглянуто метод байєсівських мереж, алогоритми побудови, критерії оцінки якості розв’язку, наведено приклад застосування. Обрано мережі Байєса через те, що вони дають змогу візуально побачити і тим самим зрозуміти складність процесу, а також дозволяють використовувати категоріальні дані для побудови моделі та формування ймовірнісного висновку на її основі.

# РОЗДІЛ 3 АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ

## 3.1 Обґрунтування вибору платформи та мови програмування

Для реалізації програмного продукту було обрано Python. Ця мова досить нова, але за останні роки досить стрімко набирає популярність. У цьому році вона ввійшла в трійку найбільш популярних. Все це тому, що вона досить легка в вивчені та має інтуїтивно зрозумілий синтаксис. Також для Python розроблено велику кількість бібліотек, які значно спрощують роботу розробникам. Існує багато реалізованих алгоритмів, що активно використовуються для машинного навчання та роботи з великими масивами даних. Саме тому ця мова є найбільш популярною серед сучасних дата-сайнтистів.

Python невибагливий до платформи і може працювати під керуванням більшості операційних систем, включаючи найбільш популярні Windows 10, Mac OS X, Android, IOS та інші.

Виходячи із задачі роботи, були використані наступні бібліотеки мови Python:

- pandas – використовується для зручного оперування масивами даних у вигляді дата фреймів;
- numpy – швидкі обчислення, операції з матрицями та інше;
- tkinter – побудова інтерфейсу користувача, додавання текстових полів, клавіш та графіків;
- sclearn – різні методи інтелектуального аналізу даних, включаючи обчислення побудовані на основі теореми Байєса;
- networkx, matplotlib – побудова різноманітних графіків та графів мережі Байєса.

## 3.2 Побудова моделі та прогнозування

Перший випадок – побудова повної мережі Байєса. Це означає, що результуюча вершина буде залежати від всіх інших, незважаючи на кореляцію між ними. Повна байєсівська мережа з результатами прогнозування тестової вибірки з 10 елементів зображена на рисунку 3.1.

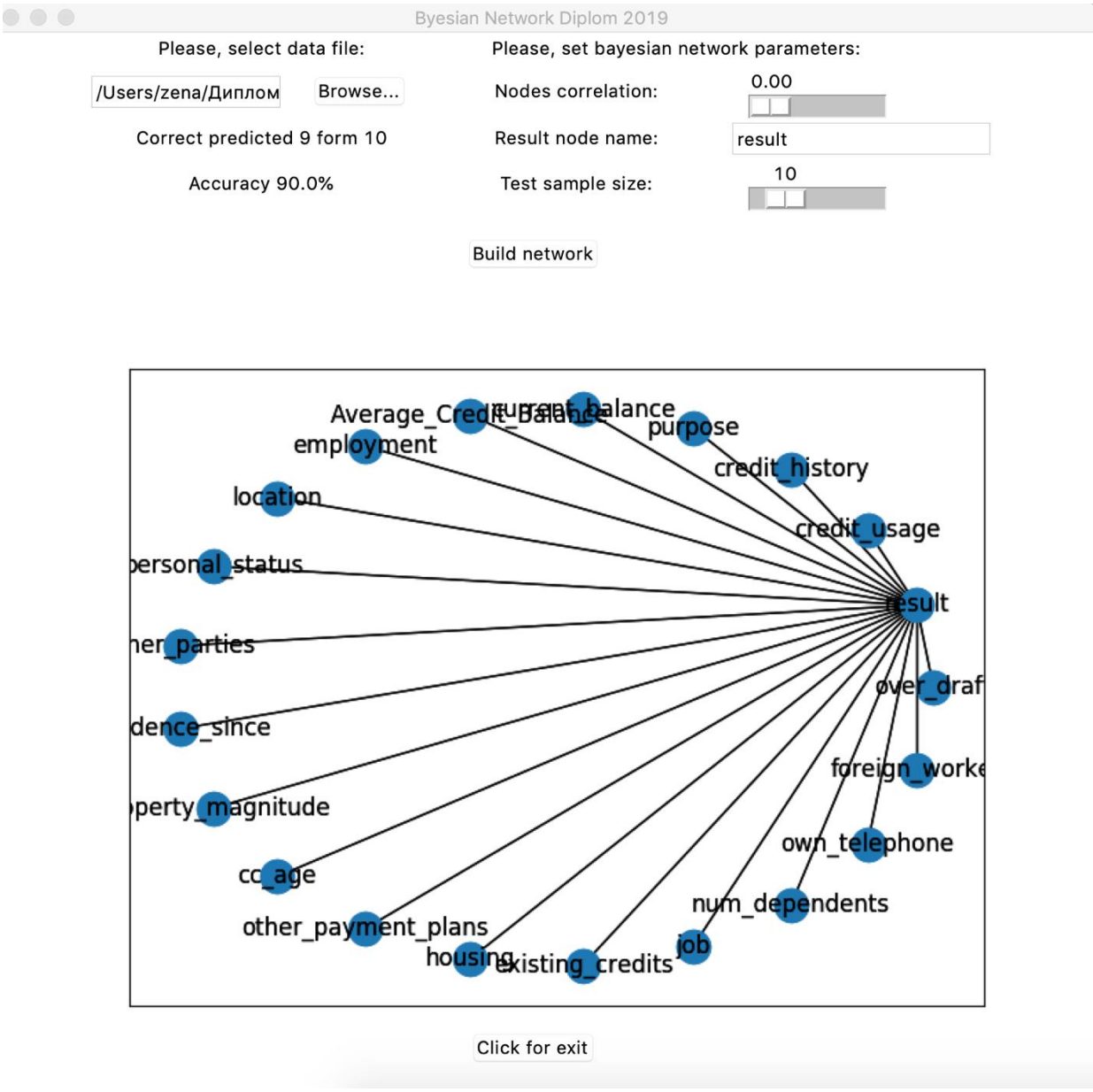



Рисунок 3.1 - Модель повної мережі Байєса.

Результати отримані при прогнозування даною моделлю відображені в таблиці 3.1.

Таблиця 3.1 – Результати прогнозування з використанням повної мережі Байєса.

| Розмір навчальної вибірки | Розмір тестової вибірки | Точність прогнозу, % |
|---|---|---|
| 990 | 10 | 90 |
| 985 | 15 | 93.333 |
| 980 | 20 | 80 |

Повна мережа Байєса витрачає досить багато ресурсів комп’ютера, що не завжди це є доречним. Тому можна зменшити кількість вершин в мережі, взявши лише ті, що корелюють між собою. Візьмемо коефіцієнт кореляції 0,1. Знайдемо вершини, які корелюють з результуючою більше ніж на заданий коефіцієнт кореляції і потім продовжимо шукати залежні вершини до знайдених і так далі, поки не залишиться вільних вершин. Результат такого моделювання зображено на рисунку 3.2.

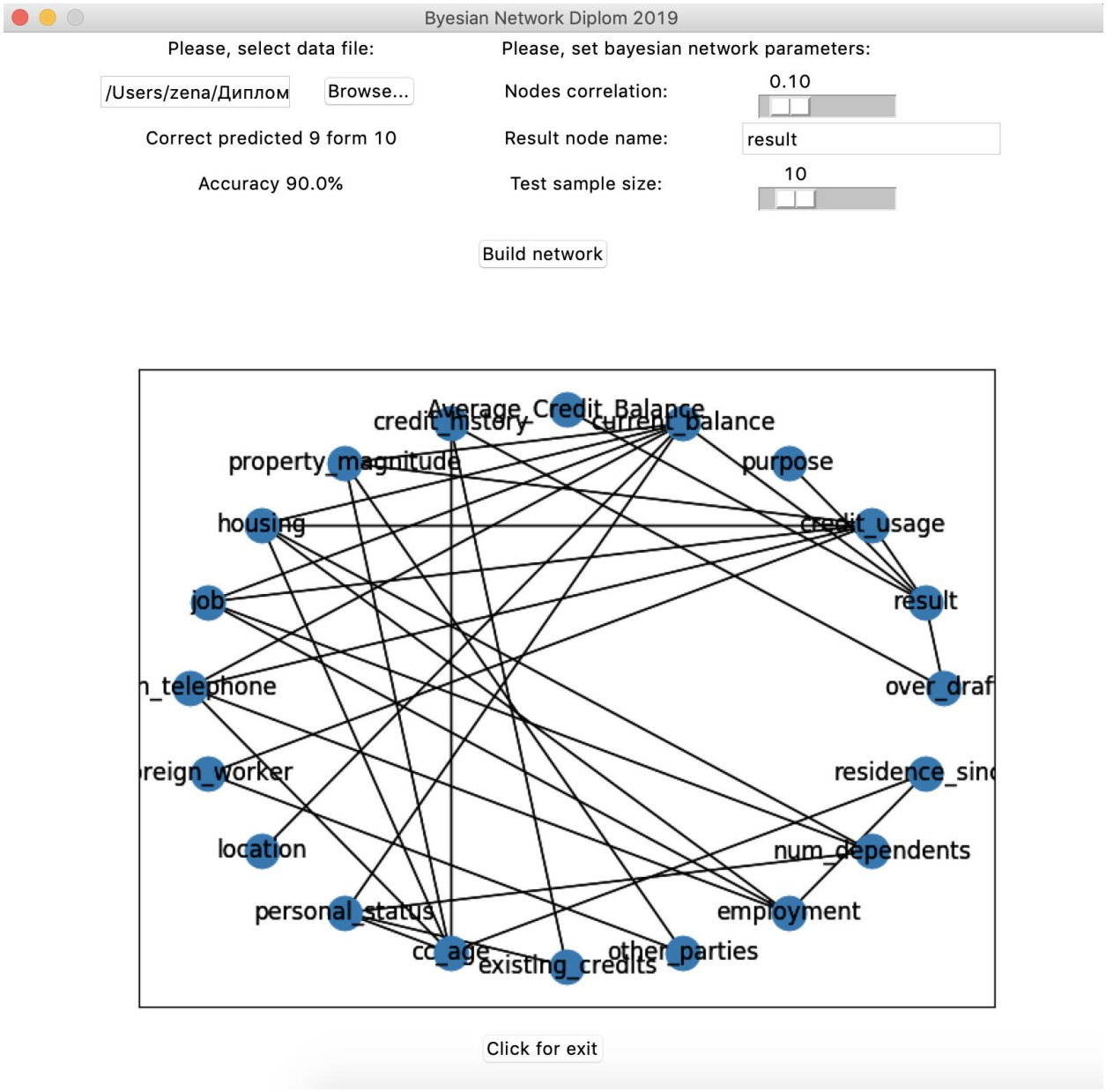



Рисунок 3.2 - Модель мережі Байєса з коефіцієнтом кореляції 0,1.

Результати отримані при прогнозування даною моделлю у вигляді мережі Байєса відображені в таблиці 3.2.

Таблиця 3.2 – Результати прогнозування мережею Байєса з коефіцієнтом кореляції 0,1.

| Розмір навчальної вибірки | Розмір тестової вибірки | Точність прогнозу, % |
|---|---|---|
| 990 | 10 | 90 |
| 985 | 15 | 80 |
| 980 | 20 | 80 |

Як видно з отриманих результатів, можна спростити мережу, при цьому не втрачаючи у точності прогнозування. Збільшивши коефіцієнт кореляції між змінними до 0.15 отримаємо модель, зображену на рисунку 3.3.

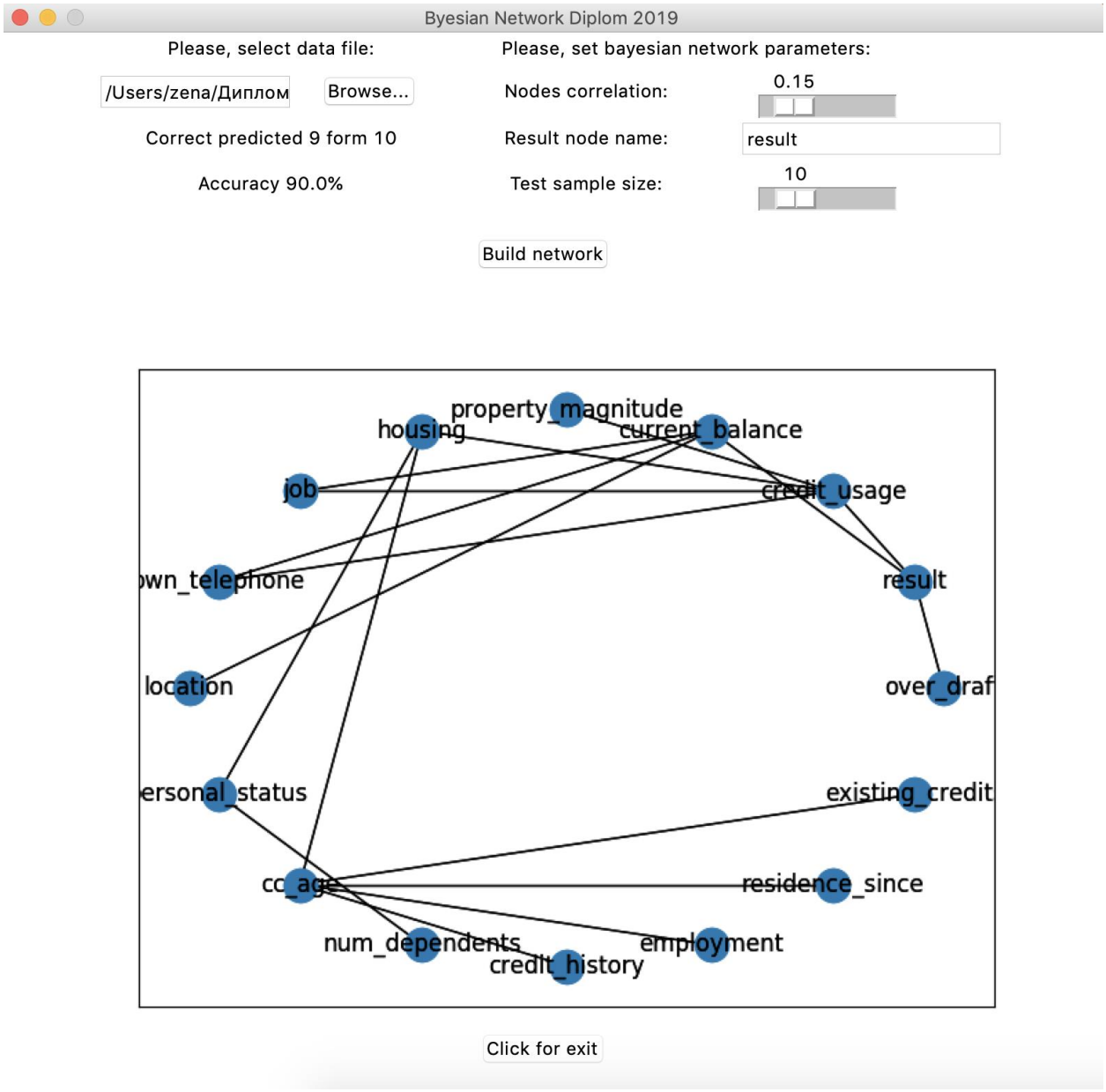



Рисунок 3.3 - Модель мережі Байєса з коефіцієнтом кореляції 0,15.

Результати такого моделювання можна спостерігати в таблиці 3.3.

Таблиця 3.3 – Результати прогнозування мережею Байєса з коефіцієнтом кореляції 0,15.

| Розмір навчальної вибірки | Розмір тестової вибірки | Точність прогнозу, % |
|---|---|---|
| 990 | 10 | 90 |
| 985 | 15 | 80 |
| 980 | 20 | 80 |

У останньому випадку точність дещо зменшилась, але не досить критично, тому продовжимо спрощувати мережу, збільшуючи коефіцієнт кореляції до 0,2. Отримаємо модель на рисунку 3.4.

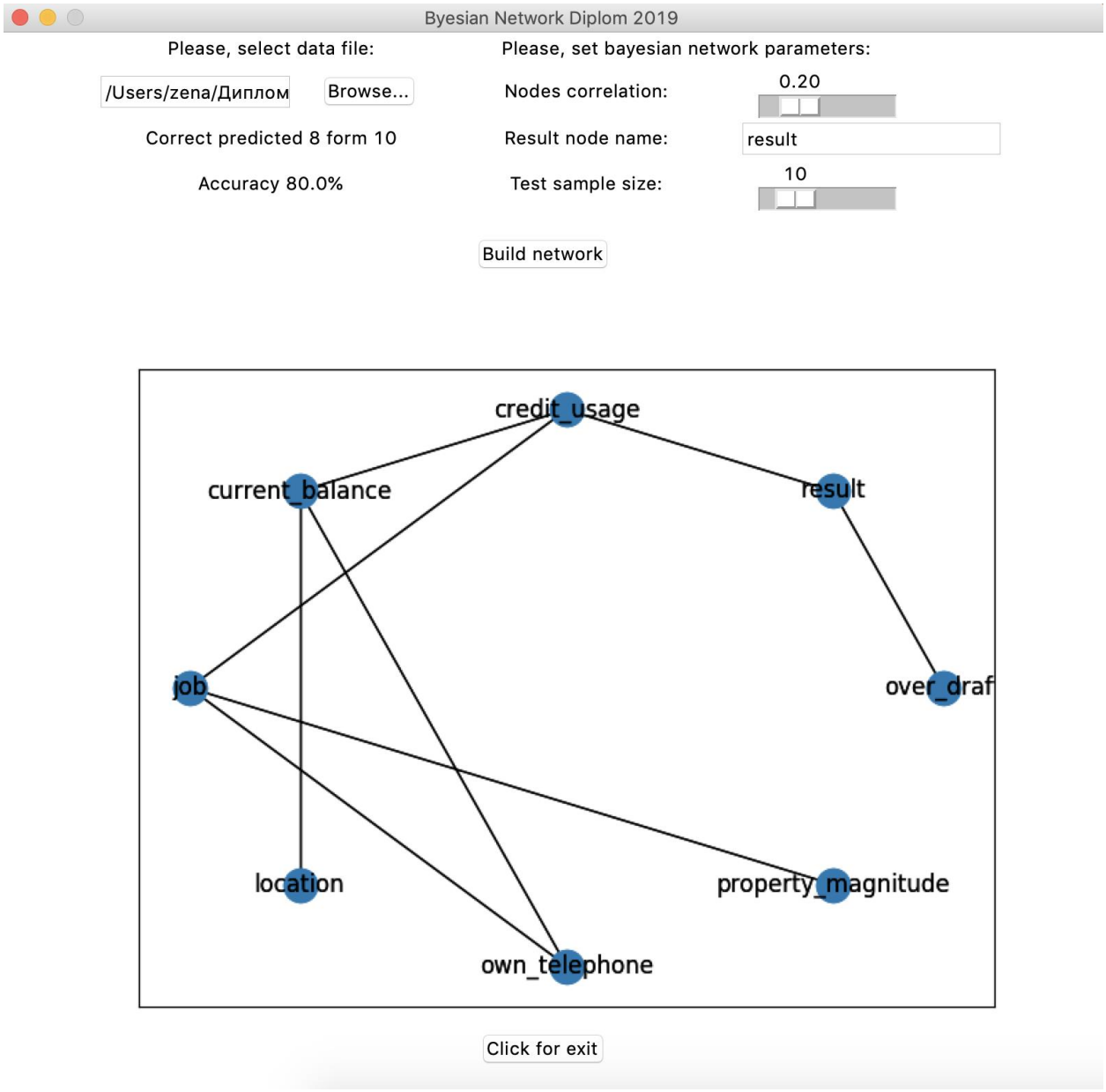



Рисунок 3.4 - Модель мережі Байєса з коефіцієнтом кореляції 0,2.

Результати прогнозування такою мережею у таблиці 3.4.

Таблиця 3.4 – Результати прогнозування мережею Байєса з коефіцієнтом кореляції 0,2.

| Розмір навчальної вибірки | Розмір тестової вибірки | Точність прогнозу, % |
|---|---|---|
| 990 | 10 | 80 |
| 985 | 15 | 73,33 |
| 980 | 20 | 75 |

## 3.3 Порівняльний аналіз отриманих результатів

Побудувавши різні моделі байєсівських мереж, змінюючи порогове значення кореляції між змінними та розмір тестової вибірки, отрмали результати, відображені в таблиці 3.5.

Таблиця 3.5 – Отримані результати точності прогнозування (у %).

| Розмір тестової вибірки | Кофіцієнт кореляції | | | |
|---|---|---|---|---|
| | 0 | 0,1 | 0,15 | 0,2 |
| 10 | 90 | 90 | 90 | 80 |
| 15 | 93.333 | 80 | 80 | 73,33 |
| 20 | 80 | 80 | 80 | 75 |

Як видно з таблиці найкращий прогноз дає повна байєсівська мережа, що і зрозуміло, оскільки в ній враховується найбільша кількість факторів, що впливають на результуючу змінну. Але не завжди зручно використовувати повну мережу, оскільки вона є досить затратною до ресурсів, тому потрібно було проаналізувати як поводять себе показники точності прогнозу при

зменшені об’ємів мережі. Враховуючи знайдені результати найбільш оптимальною є мережа з коефіцієнтом кореляції 0,15, оскільки вона використовує значно менше даних, ніж повна мережа Байєса, але при цьому показує таку ж саму точність на тестових наборах розміром 10 та 20 елементів.

Зменшення байєсівської мережі відбувалось за рахунок введення деякого порогового коефіцієнту кореляції і відкидання тих факторів, для яких значення коефіцієнту кореляції менше за задане. Алгоритм, який використовувався для оптимізації структури мережі можна записати так:

1. Визначаємо коефіцієнт кореляції $k$ та результуючу змінну, для якої будемо виконувати прогнозуванн.
2. Шукаємо змінні, які корелюють із заданою з коефіцієнтом(по модулю) більшим або рівним за $k$.
3. З’єднуємо результуючу змінну із набором знайдених факторів, оскільки вони є найбільш впливовими.
4. Для кожної змінної із знайденого набору виконуємо крок 2 і 3. Повторюємо поки не припинеться додавання нових змінних.

## 3.4 Висновки до розділу 3

У розділі описано аналіз результатів роботи програми для моделювання байєсівських мереж, написаної на мові програмування Python, яка є найбільш популярною мовою для аналізу даних на сьогоднішній день. Наведено приклади декількох мереж Байєса з різними параметрами та показано як впливає зменшення об’єму мережі на результати прогнозування. В залежності від цілей, необхідної точності та ресурсів можна обрати відповідну модель для

прогнозування нелінійних нестаціонарних процесів, поданих у вигляді категоріальних даних.

# РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі проводиться оцінка основних характеристик програмного продукту, розробленого в рамках дипломної роботи. Програмний продукт написаний на мові програмування Python 3.6 у середовищі розробки Jupyter Notebook.

В даному розділі проводиться аналіз варіантів реалізації модулю з метою вибору оптимального, з економічної точки зору. А саме проводиться функціонально-вартісний аналіз (ФВА).

Функціонально-вартісний аналіз — це метод комплексного техніко-економічного дослідження об’єкта з метою розвитку його корисних функцій при оптимальному співвідношенні між їхньою значимістю для споживача і витратами на їхнє здійснення.

Є одним з основних методів оцінки вартості науково-дослідної роботи, оскільки ФВА враховує як технічну оцінку продукту, що розробляється, так і економічну частину розробки.

Крім того, даний метод дозволяє вибрати оптимальний, як з погляду розробника, так і з точки зору покупця варіант розв’язання будь-якої задачі, а також дозволяє оптимізувати витрати й час виконання робіт.

Мета ФВА полягає у забезпеченні правильного розподілу ресурсів, виділених на виробництво продукції або надання послуг, на прямі та непрямі витрати. У даному випадку – аналізу функцій програмного продукту й виявлення усіх витрат на реалізацію цих функцій.

Фактично цей метод працює за таким алгоритмом:

а) визначається послідовність функцій, необхідних для виробництва продукту. Спочатку – всі можливі, потім вони розподіляються по двом групам:

ті, що впливають на вартість продукту і ті, що не впливають. На цьому ж етапі оптимізується сама послідовність скороченням кроків, що не впливають на цінність і відповідно витрат.

б) для кожної функції визначаються повні річні витрати й кількість робочих часів.

в) для кожної функції на основі оцінок попереднього пункту визначається кількісна характеристика джерел витрат.

г) після того, як для кожної функції будуть визначені їх джерела витрат, проводиться кінцевий розрахунок витрат на виробництво продукту.

## 4.1 Постановка задачі техніко-економічного аналізу

У роботі застосовується метод ФВА для проведення техніко-економічний аналізу розробки.

Відповідно цьому варто обирати і систему показників якості програмного продукту.

Технічні вимоги до продукту наступні:

– програмний продукт повинен функціонувати на персональних комп’ютерах із стандартним набором компонент;

– забезпечувати високу швидкість обробки великих об’ємів даних у реальному часі;

– забезпечувати зручність і простоту взаємодії з користувачем або з розробником програмного забезпечення у випадку використовування його як модуля;

– передбачати мінімальні витрати на впровадження програмного продукту.

### 4.1.1 Обґрунтування функцій програмного продукту

Головна функція $F_0$ – розробка програмного продукту, який аналізує процес за вхідними даними та будує його модель для подальшого прогнозування. Виходячи з конкретної мети, можна виділити наступні основні функції ПП:

$F_1$ – вибір мови програмування;

$F_2$ – вибір оптимального середовища розробки;

$F_3$ – інтерфейс користувача.

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція $F_1$:

а) мова програмування Eviews;

б) мова програмування Python.

Функція $F_2$:

а) IDE;

б) PyCharm.

Функція $F_3$:

а) додаток в браузері;

б) консольний додаток.

### 4.1.2 Варіанти реалізації основних функцій

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 4.1). На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (таблиця 4.1).

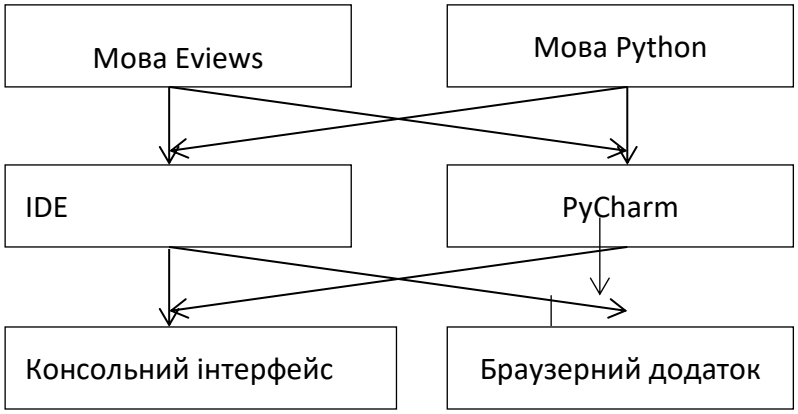



Рисунок 4.1 – Морфологічна карта

Морфологічна карта відображує всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

Таблиця 4.1 – Позитивно-негативна матриця

| Основні функції | Варіанти реалізації | Переваги | Недоліки |
|---|---|---|---|
| F1 | А | Повністю готове середовище для роботи | Не має безкоштовної ліцензії |
| | Б | Простота реалізації в рамках даної задачі | Швидкодія |
| F2 | А | Виділення синтаксису | Довгий процес виправлення помилок |

| | | | |
|---|---|---|---|
| | Б | Можливість виконувати програму блоками крок за кроком | Займає значний обсяг пам’яі на диску |
| F3 | А | Не вимогливий до потужностей комп’ютера | Не 'user-friendly' |
| | Б | Зручність для пересічного користувача | Повільніше працює |

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція F1:

Оскільки в рамках даної задачі налаштування Python не займають багато часу, обираємо варіант Б.

Функція F2:

При роботі з даними необхідно постійно виправляти та доповнювати код, тому вибираємо варіант Б.

Функція F3:

Оскільки, щодо інтерфейсу програмного продукту не має вишуканих вимог (ПП є виключно робочою частиною, в подальшому планується абсолютна автоматизація), то обидва варіанти А і Б влаштовують.

Таким чином, будемо розглядати такі варіанти реалізації ПП:

1. F1Б – F2Б – F3А
2. F1Б – F2Б – F3Б

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

## 4.2 Обґрунтування системи параметрів ПП

### 4.2.1 Опис параметрів

На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

– X1 – швидкодія мови програмування;

– X2 – час обробки даних;

– X3 – потенційний об’єм програмного коду.

X1: Відображає швидкодію операцій залежно від обраної мови програмування.

X2: Відображає час, який витрачається на дії.

X3: Показує розмір програмного коду який необхідно створити безпосередньо розробнику.

### 4.2.2 Кількісна оцінка параметрів

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у табл. 4.2.

Таблиця 4.2 – Основні параметри ПП

| Назва Параметра | Умовні позначення | Одиниці виміру | Значення параметра | | |
|---|---|---|---|---|---|
| | | | гірші | середні | кращі |
| Швидкодія мови програмування | X1 | Оп/мс | 19000 | 11000 | 2000 |
| Час обробки запитів | X2 | мс | 1000 | 420 | 60 |
| Потенційний об’єм прогр. коду | X3 | кількість строк коду | 350 | 200 | 100 |

За даними таблиці 4.2 будуються графічні характеристики параметрів – рис. 4.2 – рис. 4.4.

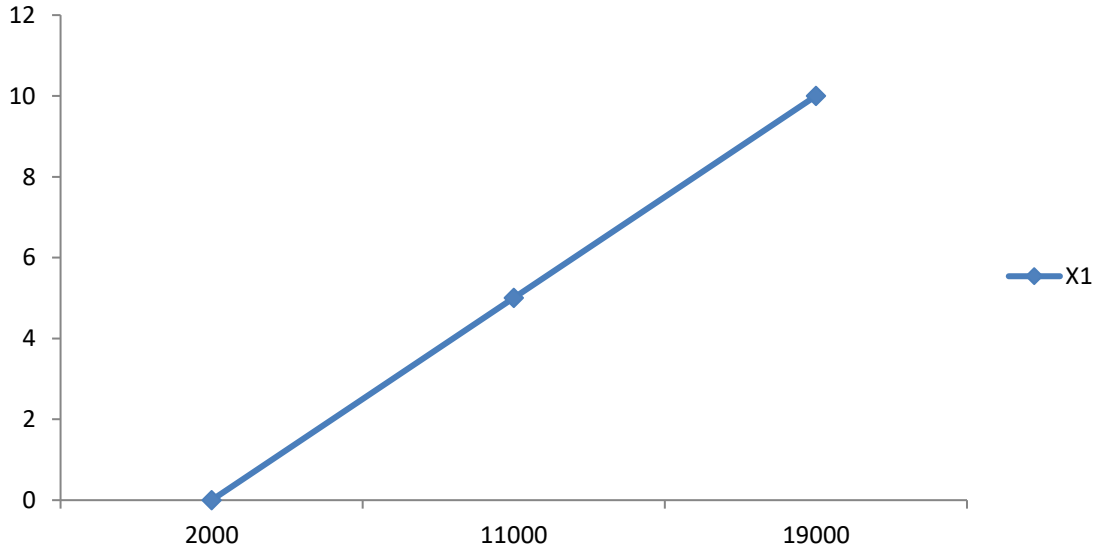



Рисунок 4.2 – X1, швидкодія мови програмування

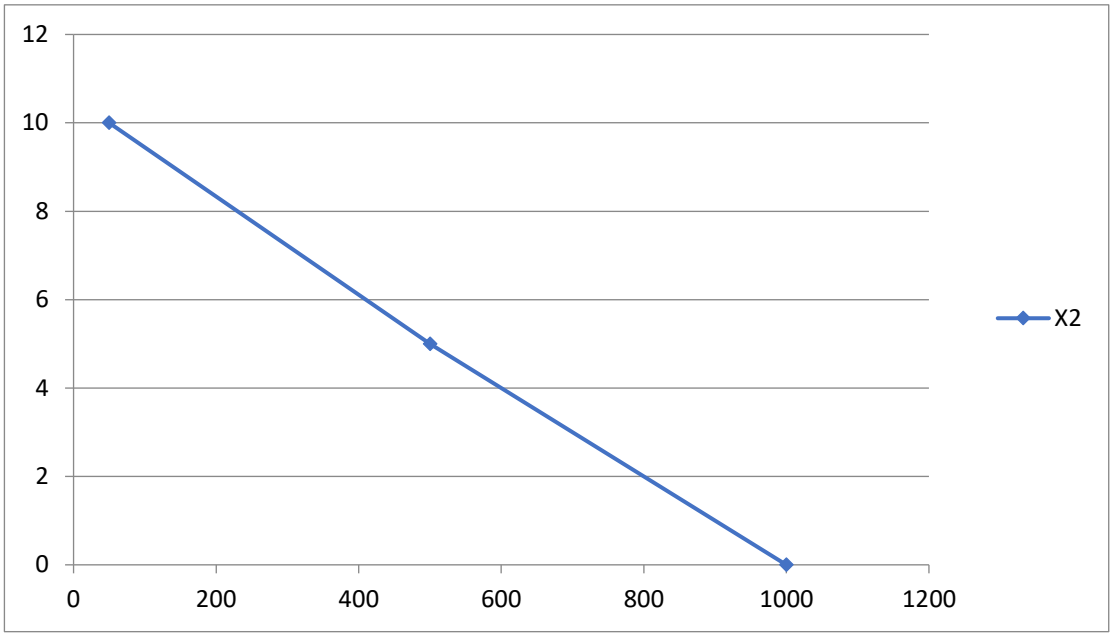



Рисунок 4.3 – X2, час виконання запитів користувача

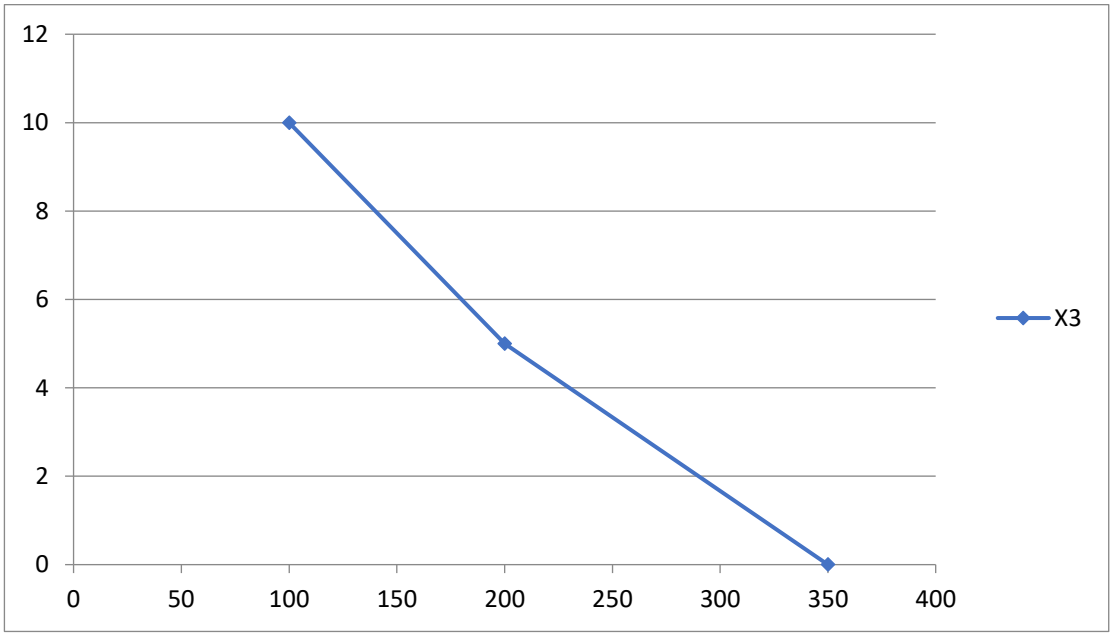



Рисунок 4.4 – X3, потенційний об’єм програмного коду

### 4.2.3 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

– визначення рівня значимості параметра шляхом присвоєння різних рангів;

– перевірку придатності експертних оцінок для подальшого використання;

– визначення оцінки попарного пріоритету параметрів;

– обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 – Результати ранжування параметрів

| Позначення параметра | Назва параметра | Одиниці виміру | Ранг параметра за оцінкою експерта | | | | | | | Сума рангів $R_i$ | Відхилення $\Delta_i$ | $\Delta_i^2$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | | |
| X1 | Швидкодія мови програмування | Оп/мс | 2 | 2 | 2 | 1 | 2 | 2 | 1 | 12 | 0,33 | 0,109 |
| X2 | Час обробки запитів | Мс | 2 | 2 | 1 | 2 | 1 | 2 | 2 | 12 | -14,67 | 215,2 |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ХЗ | Потенційний об’єм програмного коду | кількість строк коду | 2 | 2 | 3 | 3 | 3 | 2 | 3 | 18 | 14,33 | 214,92 |
| | Разом | | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 42 | 0 | 430,229 |

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 42, \quad (4.1)$$

де N – число експертів, n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 26{,}67 \quad (4.2)$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T \quad (4.3)$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 430{,}229 \quad (4.4)$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3-n)} = \frac{12 \cdot 430{,}229}{7^2(3^3-3)} = 4{,}39 > W_{норм} = 0{,}67 \quad (4.5)$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 4.4.

Таблиця 4.4 – Попарне порівняння параметрів

| Параметри | Експерти | | | | | | | Кінцева оцінка | Числове значення |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | |
| X1 і X2 | > | > | > | > | > | > | > | > | 1,5 |
| X1 і X3 | < | < | < | < | < | < | < | < | 0,5 |
| X2 і X3 | < | < | < | < | < | < | < | < | 0,5 |

Числове значення, що визначає ступінь переваги і–го параметра над j–тим, $a_{ij}$ визначається по формулі:

$$a_{ij} = \begin{cases} 1.5, \text{при Xi} > \text{Xj} \\ 1.0, \text{при Xi} = \text{Xj} \\ 0.5, \text{при Xi} < \text{Xj} \end{cases} \quad (4.6)$$

З отриманих числових оцінок переваги складемо матрицю $A = \| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{вi}$ за наступними формулами:

$$K_{вi} = \frac{b_i}{\sum_{i=1}^{n} b_i}, \quad (4.7)$$

де $b_i = \sum_{i=1}^{N} a_{ij}$.

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%).

На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{вi} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \text{де } b_i' = \sum_{i=1}^{N} a_{ij} b_j. \quad (4.8)$$

Як видно з таблиці 5.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 4.5 – Розрахунок вагомості параметрів

| Параметри $x_i$ | Параметри $x_j$ | | | Перша ітер. | | Друга ітер. | | Третя ітер. | |
|---|---|---|---|---|---|---|---|---|---|
| | X1 | X2 | X3 | $b_i$ | $K_{вi}$ | $b_i^1$ | $K_{вi}^1$ | $b_i^2$ | $K_{вi}^2$ |
| X1 | 1,0 | 1,5 | 0,5 | 3.0 | 0.333 | 8.0 | 0.32 | 21.25 | 0.31 |
| X2 | 0,5 | 1,0 | 0,5 | 2.0 | 0.222 | 5.5 | 0.22 | 15.25 | 0.223 |
| X3 | 1,5 | 1,5 | 1,0 | 4.0 | 0. 445 | 11.5 | 0.46 | 31.75 | 0.465 |
| Всього: | | | | 9 | 1 | 25 | 1 | 68.25 | 1 |

## 4.3 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо. Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 4.6):

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j}, \quad (4.9)$$

де n – кількість параметрів;

$K_{вi}$– коефіцієнт вагомості і–го параметра;

$B_i$ – оцінка і–го параметра в балах.

Таблиця 4.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

| Основні функції | Варіант реалізації функції | Параметри $x_i$ | Абсолютне значення параметра | Бальна оцінка параметра | Коефіцієнт вагомості параметра | Коефіцієнт рівня якості |
|---|---|---|---|---|---|---|
| F1 | б) | Xl | 11000 | 7 | 0,312 | 2,184 |
| | | X3 | 200 | 5 | 0,465 | 2,325 |
| F2 | б) | X2 | 100 | 3 | 0,223 | 0,669 |
| | | X3 | 200 | 7,5 | 0,465 | 3,488 |
| F3 | б) | X2 | 100 | 3,5 | 0,223 | 0,78 |
| | | X3 | 350 | 1 | 0,465 | 0,465 |
| | а) | X3 | 200 | 1 | 0,465 | 0,465 |

За даними з таблиці 4.6 за формулою

$$K_K = K_{ТУ}[F_{1k}] + K_{ТУ}[F_{2k}] + \ldots + K_{ТУ}[F_{zk}], \quad (4.10)$$

визначаємо рівень якості кожного з варіантів:

$K_{K1}$ = 2,184+2,325+0,669+3,488+0,78+0,465 = 9,911

$K_{K2}$ = 2,184+2,325+0,669+3,488+0,465 = 9,131

Як видно з розрахунків, кращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

## 4.4 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки;

Але варіант II реалізації програмного забезпечення включає ще одне завдання:

3. Написання алгоритму збереження інформації у вигляді компонента, зручного для візуалізації.

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б, завдання 3 до группи Г. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3. Завдання 3 відноситься за складністю до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється як

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_М \cdot K_{СТ} \cdot K_{СТ.М}, \quad (4.11)$$

де $T_P$ – трудомісткість розробки ПП;

$K_П$ – поправочний коефіцієнт;

$К_{СК}$ – коефіцієнт на складність вхідної інформації; $К_М$ – коефіцієнт рівня мови програмування;

$К_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$К_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $Т_Р$ =90 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $К_П$ = 1.7. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для завдань рівний 1: $К_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ}$ = 0.8. Тоді, за формулою 5.1, загальна трудомісткість програмування першого завдання дорівнює:

$$Т_1 = 90 \cdot 1.7 \cdot 0.8 = 122.4 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $Т_Р$ =27 людино-днів, $К_П$ =0.9, $К_{СК}$ = 1, $К_{СТ}$ =0.8:

$$Т_2 = 27 * 0.9 * 0.8 = 19.44 \text{ людино-днів.}$$

Для третього завдання (використовується алгоритм третьої групи складності, ступінь новизни Г):

$$Т_р = 15 \text{ людино-днів;} \quad К_п = 0,6; \quad К_{ст} = 1; \quad Т_З = 15 * 0,6 * 1 = 9.$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$Т_I = (122.4 + 19.44) * 15 = 2127,6 \text{ людино-годин;}$$

$$Т_{II} = (122.4 + 19.44 + 9) * 15 = 2262,6 \text{ людино-годин;}$$

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 14000 грн., один спеціаліст по цифровій обробці сигналів з окладом 22000грн. Визначимо зарплату за годину за формулою:

$$СЧ = \frac{М}{T_m \cdot t} грн., \quad (4.12)$$

де М – місячний оклад працівників; $T_m$ – кількість робочих днів тиждень; t – кількість робочих годин в день.

$$СЧ = \frac{14000 + 14000 + 22000}{3 \cdot 21 \cdot 15} = 52{,}91 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою

$$СЗП = С_ч \cdot T_i \cdot КД, \quad (4.13)$$

де $С_ч$ – величина погодинної оплати праці програміста; $T_i$ – трудомісткість відповідного завдання; $К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП} = 52{,}91 \cdot 2127{,}6 \cdot 1.2 = 135085{,}58$ грн.

II. $С_{ЗП} = 52{,}91 \cdot 2262{,}6 \cdot 1.2 = 143657{,}00$ грн.

Відрахування на єдиний соціальний внесок в залежності від групи професійного ризику (II клас) становить 22%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.3677 = 135085{,}58 \cdot 0.22 = 29718{,}83$ грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.3677 = 143657,00 \cdot 0.22 = 31604,54$ грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Так як одна ЕОМ обслуговує одного програміста з окладом 14000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 12000 \cdot 0,2 = 28800 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$С_{ЗП} = С_Г \cdot (1 + К_З) = 28800 \cdot (1 + 0.2) = 34560 \text{ грн.}$$

Відрахування на єдиний соціальний внесок:

$$С_{ВІД} = С_{ЗП} \cdot 0.3677 = 34560 \cdot 0,22 = 7603,20 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 8000 грн.

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 8000 = 2300 \text{ грн.},$$

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача; $К_А$– річна норма амортизації; $Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 8000 \cdot 0.05 = 460 \text{ грн.},$$

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{ЕФ} =(Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В \quad (4.14)$$

$$Т_{ЕФ} = (365 – 104 – 8 – 16) \cdot 8 \cdot 0.9 = 1706.4 \text{ годин},$$

де $Д_К$ – календарна кількість днів у році; $Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів; $Д_Р$ – кількість днів планових ремонтів устаткування; t –кількість робочих годин в день; $К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} =1706,4 \cdot 0,156 \cdot 0,9733 \cdot 2,7515 = 712,88 \text{ грн.},$$

де $N_С$ – середньо-споживча потужність приладу; $К_З$– коефіцієнтом зайнятості приладу; $Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$С_Н = Ц_{ПР} \cdot 0.67 = 8000 \cdot 0,67 =5360 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} =С_{ЗП}+ С_{ВІД}+ С_А + С_Р+ С_{ЕЛ} + С_Н \quad (4.15)$$

$$С_{ЕКС} = 34560+ 7603,20 + 2300 + 460 + 712,88 + 5360 = 50996,08 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$C_{М\text{-}Г} = C_{ЕКС} / T_{ЕФ} = 50996,08/1706,4 = 29,885 \text{ грн/час.}$$

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$C_М = C_{М\text{-}Г} \cdot T \quad (4.15)$$

I. $C_М = 29,885 \cdot 2127,6 = 63583,326$ грн.;

II. $C_М = 29,885 \cdot 2262,6 = 67617,801$ грн.;

Накладні витрати складають 67% від заробітної плати:

$$C_Н = C_{ЗП} \cdot 0,67 \quad (4.16)$$

I. $C_Н = 135085,58 \cdot 0,67 = 90507,34$ грн.;

II. $C_Н = 143657,00 \cdot 0,67 = 96250,19$ грн.;

Отже, вартість розробки ПП за варіантами становить:

$$C_{ПП} = C_{ЗП} + C_{ВІД} + C_М + C_Н \quad (4.17)$$

I. $C_{ПП} = 135085,58 + 49670,97 + 63583,326 + 90507,34 = 338847,216$ грн.;

II. $C_{ПП} = 143657,00 + 52822,68 + 67617,801 + 96250,19 = 360347,671$ грн.;

## 4.5 Вибір кращого варіанта ПП за техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕРj} = К_{Кj} / С_{Фj}, \quad (4.18)$$

$$К_{ТЕР1} = 9,911 / 344325,79 = 2,88 \cdot 10^{-5};$$
$$К_{ТЕР2} = 9,131 / 366173,97 = 2,49 \cdot 10^{-5};$$

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 2,88 \cdot 10^{-5}$.

## 4.6 Висновки до розділу 4

В даному розділі проведено повний функціонально-вартісний аналіз ПП, який було розроблено в рамках дипломної роботи. Процес аналізу можна умовно розділити на дві частини.

В першій з них проведено дослідження ПП з технічної точки зору: було визначено основні функції ПП та сформовано множину варіантів їх реалізації; на основі обчислених значень параметрів, а також експертних оцінок їх важливості було обчислено коефіцієнт технічного рівня, який і дав змогу визначити оптимальну з технічної точки зору альтернативу реалізації функцій ПП.

Другу частину ФВА присвячено вибору із альтернативних варіантів реалізації найбільш економічно обґрунтованого. Порівняння запропонованих варіантів реалізації в рамках даної частини виконувалось за коефіцієнтом ефективності, для обчислення якого були обчислені такі допоміжні параметри, як трудомісткість, витрати на заробітну плату, накладні витрати.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР}$ $2{,}88 \cdot 10^{-5}$.

Цей варіант реалізації програмного продукту має такі параметри:

- мова програмування – Python;
- використання PyCharn IDE;
- браузерний інтерфейс.

Даний варіант виконання програмного комплексу дає користувачу відмінний функціонал, швидкодію і робить простішим виконання завдання.

# ВИСНОВКИ

Величезна кількість інформації, яка потребує адекватної обробки та прийняття рішень на основі результатів цієї обробки, накопичується у всьому світі. Методи інтелектуального аналізу даних (IDA), які включають байєсовські мережі (BNs), дають можливість автоматичного пошуку принципів, характерних для багатовимірних даних. З кожним днем все більше і більше людей розуміють, що інтелектуальний аналіз даних – тренд, актуальність якого підтверджується щодня. Вміння оперувати великими масивами даних, застосовувати до них методи аналізу – запорука успіху будь-якого бізнесу в сучасному інформатизованому суспільстві.

Більшість інструментів для інтелектуального аналізу даних лежать в основі двох методів: машинного навчання та візуалізації даних. Байєські мережі об'єднують обидва ці методи.

У роботі проаналізовано методи, за допомогою яких можна моделювати фінансово-економічні нелінійні нестаціонарні процеси та забезпечувати можливість оцінювання короткострокових прогнозів фінансових, економічних та інших показників. Виконано огляд основних методів та алгоритмів інтелектуального аналізу даних, моделей, побудованих на основі авторегресії та методу ковзного вікна.

Також було вивчено та протестовано готові програмні продукти для побудови і обрахунку ймовірності тієї чи іншої події на основі мереж Байєса. Найяскравішим прикладом такого застосунку є GeNIe Modeller, де в зручному графічному інтерфейсі користувач має змогу створювати вершини мережі, з’єднувати їх між собою, та надавати ймовірнісні показники кожному з можливих станів вершин.

Реалізовано програмний продукт для побудови мережі Байєса, яка спроможна ймовірнісно оцінювати операційні ризики кредитування.

Мережа побудована на основі аналізу набору даних тисячі клієнтів німецького банку, що містить 20 факторів, які впливаються на рішення стосовно видачі кредиту.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Згуровський М. З., Бідюк П. І., Терентьєв О. М. Байєсівські мережі в системах підтримки приийняття рішень / Київ: ТОВ Видавниче Підприємство «Едельвейс», 2015. 300 с.

2. Бідюк П.І., Коршевнюк Л.О. Проектування комп'ютерних інформаційних систем підтримки приийняття рішень: Навчальний посібник. Київ: ННК «ІПСА» НУТУ «КПІ», 2010. 340 с.

3. П. І. Бідюк, В. Д. Романенко, О. Л.Тимощук Аналіз часових рядів (навчальний посібник). К.: Політехніка, 2010. 317 с.

4. Раздел журнала: Методы обработки информации. УДК 62-50П.И. Бидюк, А.Н. Терентьев, Л.А. Коршевнюк байесовская сеть - инструмент интеллектуального анализа данных

5. Бидюк П.И., Терентьев А.Н., Гасанов А.С. Построение и методы обучения Байесовских сетей Кибернетика и системный анализ, 2005, № 4. с. 133 – 147.

6. Challis, R. E., and Kitney, R. I. (November 1991). "Biomedical signal processing (in four parts). Part 1 Time-domain methods." Medical & Biological Engineering & Computing, 28, 509-524.

7. Лукашин Ю.П. Адаптивные методы краткосрочного прогнозирования. М.: Финансы и статистика, 2003. 414 с.

8. Зельнер А. Байесовские методы в эконометрии. Москва: Статистика, 1980. 438 с.

9. Chatfield C. Time series forecasting. London: Chapman & Hall, 2000. 267 p.

10. Згуровский М.З., Подладчиков В.Н. Аналитические методы калмановской фильтрации. Київ: Наукова думка, 1995. 285 с.

11. Fryzlewicz, P., Van Bellegem, S. & von Sachs, R 2003. Forecasting non-stationary time series by wavelet process modelling. Annals of the Institute of Statistical Mathematics, 55, 737-764. Та Priestley, M.B. 1983. Spectral Analysis and Time Series. Academic Press.

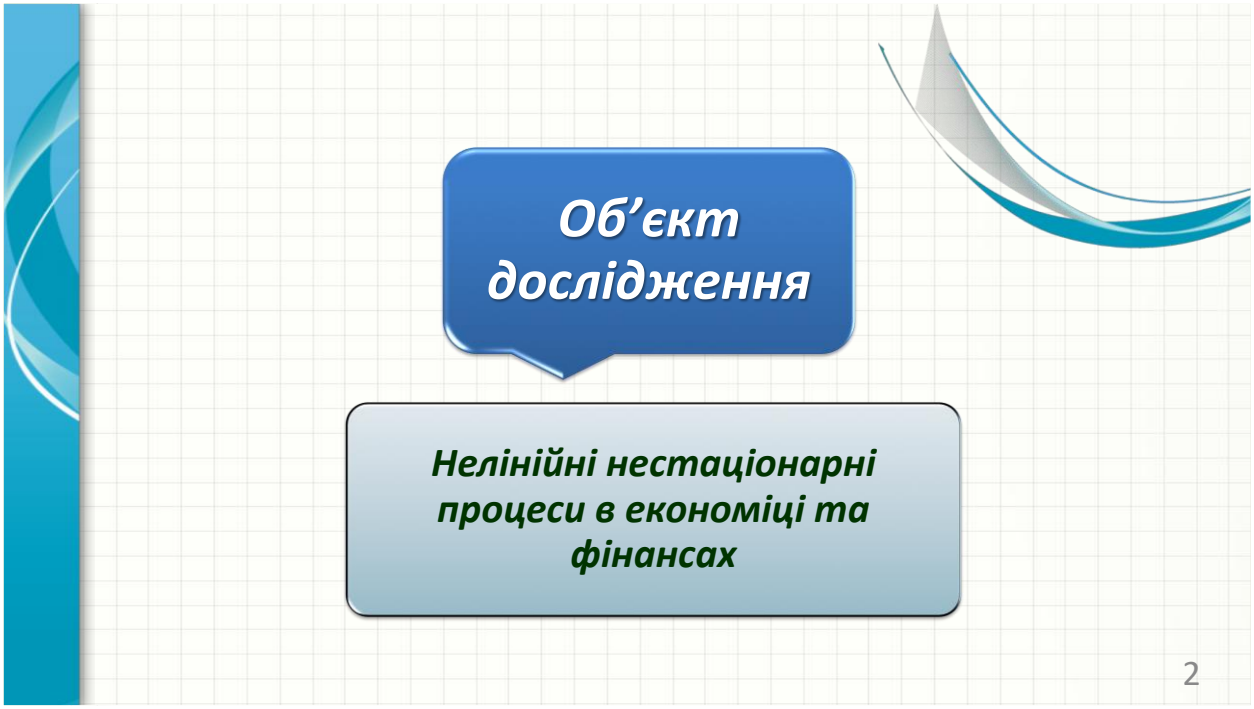
12. Конспект лекцій з теорії ймовірностей Ільєнко Андрій Борисович, Кафедра математичного аналізу та теорії ймовірностей ФМФ, Доцент, 2015

13. Посібник користувача GeNIe Modeler. URL: https://support.bayesfusion.com/docs/GeNIe.pdf

14. Introduction to Bayesian Networks URL: https://towardsdatascience.com/introduction-to-bayesian-networks-81031eeed94e

15. Bayesian networks - an introduction URL: https://www.bayesserver.com/docs/introduction /bayesian-networks

ДОДАТОК А ІЛЮСТРАТИВНИЙ МАТЕРІАЛ

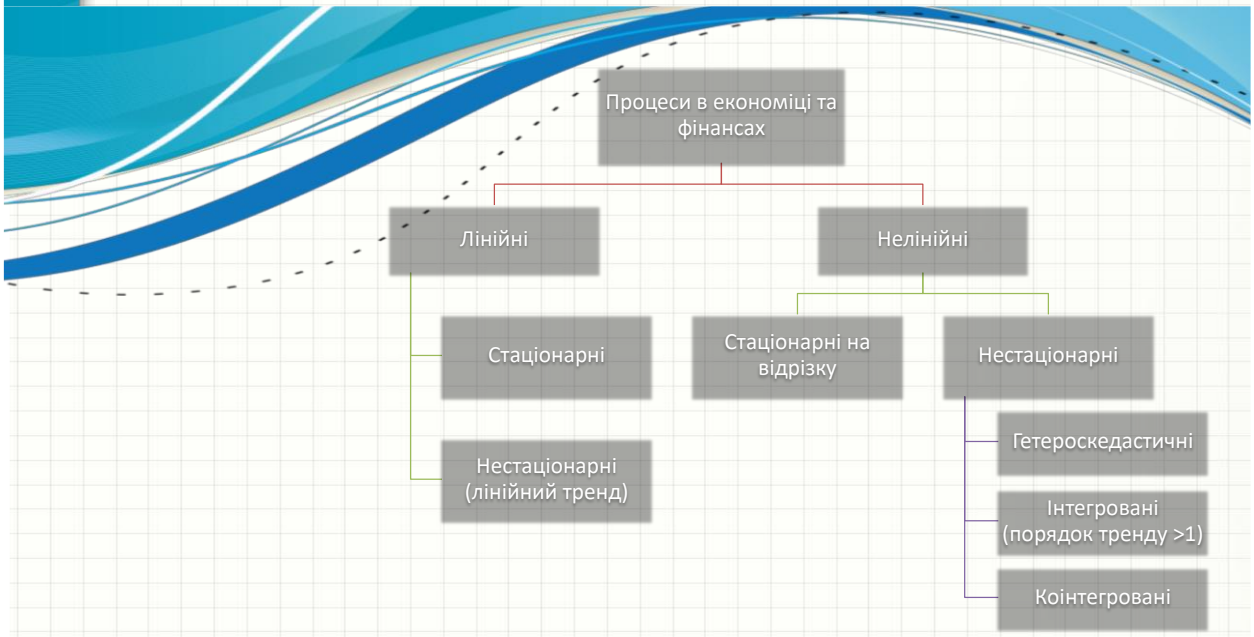




Предмет дослідження
Математичні моделі нелінійних нестаціонарних процесів, методи та методики їх побудови
3
Мета дослідження:
Побудувати адекватні моделі фінансових процесів у формі байєсівських мереж з використанням статистичних даних
4

# Постановка задачі

- *аналіз типів сучасних фінансово-економічних процесів;*
- *збір статистичних даних;*
- *вибір методу моделювання фінансових процесів у кредитуванні;*
- *побудова моделей у формі байєсівських мереж;*
- *аналіз отриманих результатів.*

5




6

# Деякі моделі нелінійних нестаціонарних процесів

| | |
|---|---|
| Авторегресія (AR) | • $y(k) = a_0 + \sum_{i=1}^{p} a_i y(k-i) + b_1 k + \cdots + b_m k^m + \varepsilon(k)$;<br>• $k = 1,\ 2,\ 3, \ldots$ – дискретний час;<br>• $t = kT_s$, де $T_s$ – час виміру спроби. |
| Авторегресія з ковзним середнім (ARMAX) | • $y(k) = a_0 + \sum_{i=1}^{p} a_i y(k-i) + \sum_{i=1}^{q} \theta_i \varepsilon(k-i) + b_1 k + \cdots + b_m k^m + \varepsilon(k)$;<br>• $k = 1,\ 2,\ 3, \ldots$ – дискретний час;<br>• $t = kT_s$, де $T_s$ – час виміру спроби. |
| Узагальнена авторегресія з умовною гетероскедастичністю (GARCH) | • $h(k) = \alpha_0 + \sum_{i=1}^{q} \alpha_i \varepsilon^2(k-i) + \sum_{i=1}^{p} \beta_i h(k-i)$. |
| Логістична регресія | • $\varphi(x(k,z)) = \frac{1}{1+e^{-x(k,z)}}$;<br>• $x(k) = \alpha_0 + \alpha_1 z_1(k) + \cdots + \alpha_m z_m(k) + \varepsilon(k)$. |
| Мережа Байєса | • Імовірнісна байєсівська мережна структура побудована за даними та / або експертними оцінками. |

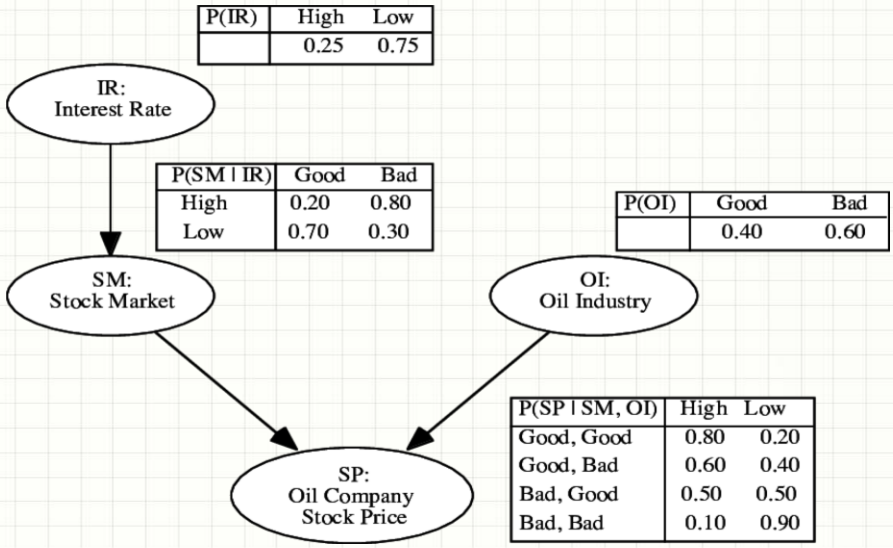
7

# Байєсівська мережа

- *Визначається як пара < G, B >, G – ациклічний граф змінних, B – множина параметрів, що визначають таблиці умовних імовірностей;*
- $P_B(X^1, \ldots, X^N) = \prod_{i=1}^{n} P_B\left(X^i | Parent(X^i)\right)$ *- формула повної ймовірності БМ;*
- *Для формування ймовірнісного висновку використовується теорема Байєса.*

8

# Байєсівська мережа




9

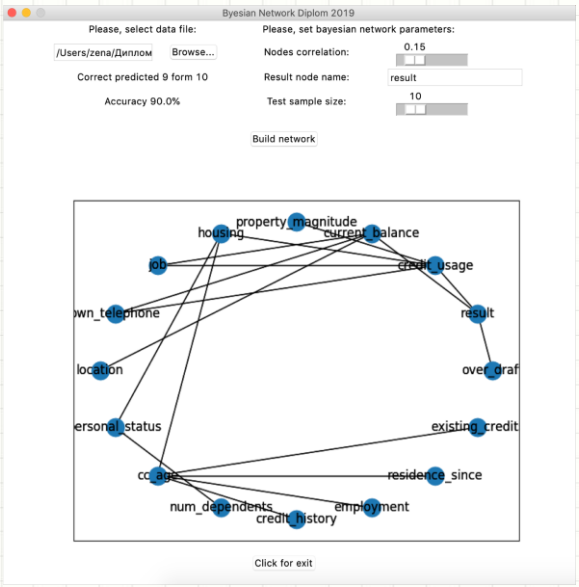
# Задача кредитування

*Для визначеності кредитоспроможності клієнта необхідно проаналізувати ряд факторів, таких як:*

- *статус поточного існуючого рахунку;*
- *тривалість кредиту у місяцях;*
- *кредитна історія;*
- *призначення кредиту;*
- *сума кредиту;*
- *ощадний рахунок/облігації;*
- *роки працевлаштування;*
- *ставка погашення у відсотках від наявного доходу;*
- *особистий статус і стать;*
- *інші сторони (поручителі);*
- *власність;*
- *вік;*
- *інші плани розстрочок;*
- *наявність житла;*
- *кількість існуючих кредитів у цьому банку;*
- *працевлаштування;*
- *кількість людей, що підлягають обслуговуванню;*
- *телефон;*
- *працевлаштування за кордоном;*

10

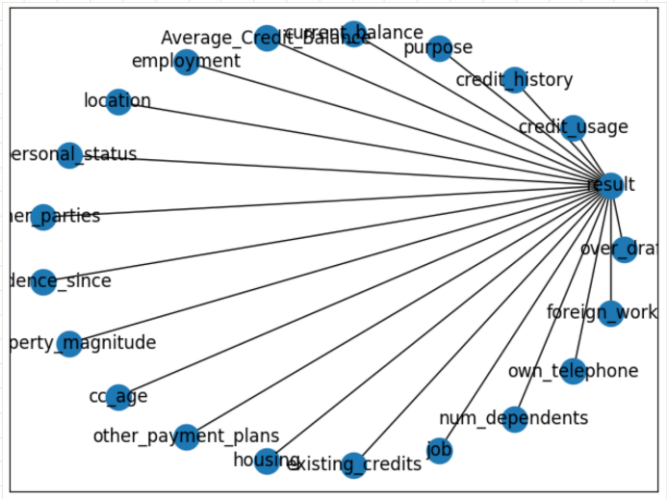
# Розроблена програма




- *Мова Python.*
- *pandas (оперування масивами даних);*
- *numpy (швидкі обчислення);*
- *tkinter (інтерфейс користувача);*
- *networkx, matplotlib (побудова графіків та графів мережі).*

11

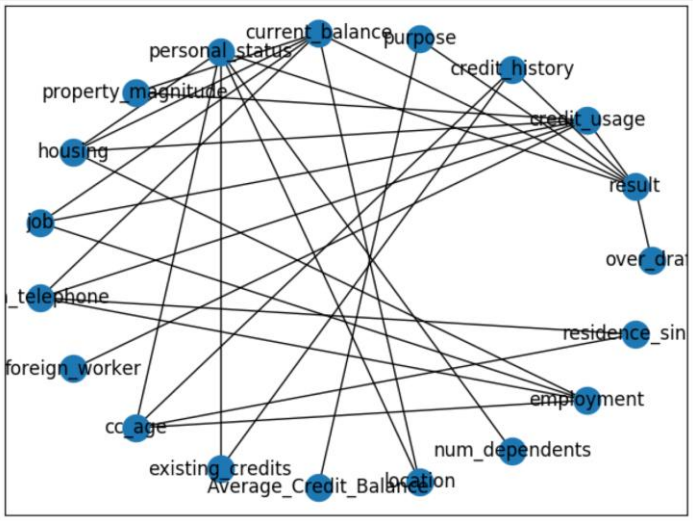
# Модель 1
# Повна байєсівська мережа




| Розмір навчальної вибірки | Розмір тестової вибірки | Точність прогнозу, % |
|---|---|---|
| 990 | 10 | 90 |
| 985 | 15 | 93.33 |
| 980 | 20 | 80 |

12

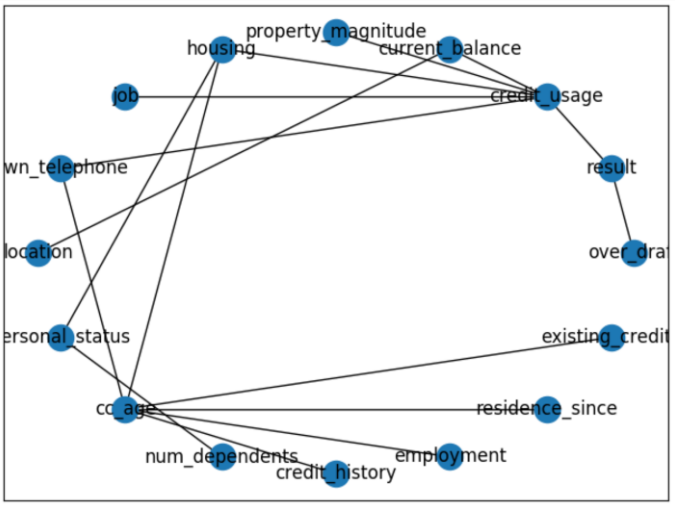
# Модель 2
# (Коефіцієнт кореляції 0.1)




| Розмір навчальної вибірки | Розмір тестової вибірки | Точність прогнозу, % |
|---|---|---|
| 990 | 10 | 90 |
| 985 | 15 | 93.33 |
| 980 | 20 | 80 |

13

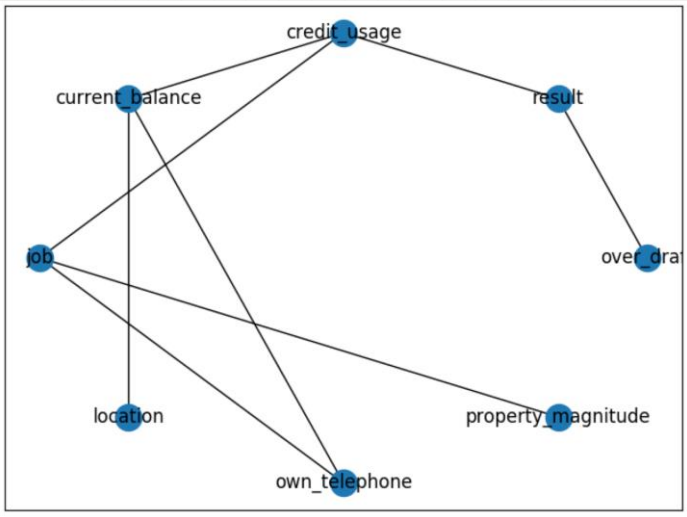
# Модель 3
# (Коефіцієнт кореляції 0.15)




| Розмір навчальної вибірки | Розмір тестової вибірки | Точність прогнозу, % |
|---|---|---|
| 990 | 10 | 90 |
| 985 | 15 | 80 |
| 980 | 20 | 80 |

14

# Модель 4
# (Коефіцієнт кореляції 0.2)




| Розмір навчальної вибірки | Розмір тестової вибірки | Точність прогнозу, % |
|---|---|---|
| 990 | 10 | 80 |
| 985 | 15 | 73.33 |
| 980 | 20 | 75 |

# Результати роботи

- *Зроблено класифікацію процесів в економіці та фінансах*
- *Вибрано деякі математичні моделі нелінійних нестаціонарних процесів, які можуть бути використані для формального опису сучасних ФЕП*
- *На основі статистичних даних кредитування клієнтів побудовано кілька моделей у формі мереж Байєса, які забезпечують високу якість прогнозування платоспроможності клієнтів банку*
- Розроблено програмний продукт на мові Python для побудови ймовірнісних моделей у формі мереж Байєса

16

# Дякую за увагу!

17

# ДОДАТОК Б ЛІСТИНГ ПРОГРАМИ

```
import matplotlib
from BayesianNetwork import *
matplotlib.use("TkAgg")
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg, NavigationToolbar2Tk
import networkx as nx
import matplotlib.pyplot as plt
from tkinter import filedialog as fd
import tkinter as tk
from tkinter import StringVar, DoubleVar, IntVar


class View(tk.Tk):
   def __init__(self, *args, **kwargs):

      tk.Tk.__init__(self, *args, **kwargs)
      tk.Tk.wm_title(self, "Byesian Network Diplom 2019")
      tk.Tk.resizable(self,0,0)
      self.corr = DoubleVar()
      self.corr.set(0.2)

      self.result = StringVar()
      self.result.set('result')
      self.test_size = IntVar()
      self.test_size.set(10)
      self.file = '/Users/zena/Дипломна робота/BN_Program/data/csv_result-credit_fruad.csv'
      self.filepath = StringVar()
      self.filepath.set(self.file)

      self.f = plt.figure(figsize=(8, 6))
      self.a = self.f.add_subplot(111)
      self.G = nx.empty_graph()

      self.num_of_correct_predicted = 0;


      """Search File textbox and button"""
      tk.Label(self, text='Please, select data file:').grid(row=0, column=1, columnspan=3)
      tk.Entry(self, textvariable=self.filepath).grid(row=1,column=1, columnspan=2)
      tk.Button(self, text="Browse...", command=self.browse).grid(row=1,column=3)
```

```
        tk.Label(self, text='Please, set bayesian network parameters:').grid(row=0, column=4,
columnspan=3)

        tk.Label(self, text='Nodes correlation:').grid(row=1, column=4, columnspan=2)
        tk.Scale(self, from_=0, to=1, resolution=0.05, orient=tk.HORIZONTAL,
variable=self.corr).grid(row=1,column=6)

        tk.Label(self, text='Result node name:').grid(row=2, column=4, columnspan=2)
        tk.Entry(self, textvariable=self.result).grid(row=2,column=6, columnspan=2)

        tk.Label(self, text='Test sample size:').grid(row=3, column=4, columnspan=2)
        tk.Scale(self, from_=2, to=50, resolution=1, orient=tk.HORIZONTAL,
variable=self.test_size).grid(row=3,column=6)

        tk.Button(self, text="Build network", command=self.build_net()).grid(row=5,column=4)

        pos = nx.circular_layout(self.G)
        nx.draw_networkx(self.G, pos=pos, ax=self.a)

        self.canvas = FigureCanvasTkAgg(self.f, master=self)
        self.canvas.draw()
        self.canvas.get_tk_widget().grid(row=6,column=0,columnspan =9, rowspan = 9)
        self.canvas._tkcanvas.grid(row=6,column=0,columnspan =9, rowspan = 9)


        tk.Button(self, text="Click for exit", command=self.destroy).grid(row=14,column=4)


        tk.Label(self, text='Forecasting results:').grid(row=2, column=1, columnspan=3)

        corr = self.num_of_correct_predicted
        total = self.test_size.get()
        tk.Label(self, text='Correct predicted ' + str(corr) + ' form ' + str(total)).grid(row=2, column=1,
columnspan=3)
        tk.Label(self, text='Accuracy ' + str(100*(corr/total)) + '%').grid(row=3, column=1,
columnspan=3)



        col_count, row_count = self.grid_size()



        for col in range(col_count):
            self.grid_columnconfigure(col, minsize=20, weight=1)
```

```
        for row in range(row_count):
            self.grid_rowconfigure(row, minsize=20, weight=1)

    def browse(self):
        filetypes =(
        ('Csv files', '*.csv'),
        ("All files", "*.*"))
        initaldir = r"C:\Users"


        self.file = fd.askopenfilename(initialdir=initaldir,
                                       filetypes=filetypes)
        self.filepath.set(self.file)

        print(self.file)

    def build_net(self):
        self.a.cla()
        self.G =
nx.from_pandas_edgelist(self.builded_network(self.filepath.get(),self.corr.get(),self.result.get(),self.te
st_size.get()), 'from', 'to')
        pos = nx.circular_layout(self.G)
        nx.draw_networkx(self.G, pos=pos, ax=self.a)

        self.canvas = FigureCanvasTkAgg(self.f, master=self)
        self.canvas.draw()


    def builded_network(self,file,corr,result,test_size):
        self.bayes_net = Network()
        self.bayes_net.setFileName(file)

        self.bayes_net.setCorrelation(corr)
        self.bayes_net.setResultNode(result)
        self.bayes_net.setSampleSetSize(test_size)
        self.bayes_net.readDataFromFile()

        graph =  self.bayes_net.getGraphDataFrame()
        self.num_of_correct_predicted = self.bayes_net.train()

        return graph


import numpy as np
import pandas as pd
from sklearn.naive_bayes import GaussianNB, ComplementNB, BernoulliNB
```

```
class Network():
    def __init__(self):
        self.nodes = {}
        self.dependenses = {}
        self.used_columns = []
        self.model = GaussianNB()
        # self.corr = 0

    def setFileName(self,file_name):
        self.file_name = file_name

    def readDataFromFile(self):
        self.df = pd.read_csv(self.file_name)
        del self.df['id']
        print(np.array(self.df[:-1].values))
        self.test_df = self.df.tail(self.n)
        self.df = self.df.head(self.df.__len__()-self.n)
        self.corr_df = self.df
        for str_column in self.corr_df.select_dtypes(exclude=[np.number]).keys():
            self.corr_df[str_column] = self.corr_df[str_column].astype('category').cat.codes
            self.test_df[str_column] = self.test_df[str_column].astype('category').cat.codes

        self.corr_df = self.corr_df.corr(method='pearson')

        with pd.option_context('display.max_rows', None, 'display.max_columns',
                               None):  # more options can be specified also
            print(self.corr_df)


        # for column in self.df.columns.values:
        #     if (column != 'id'):
        #         self.addNode(column)

    def getDataFrame(self):
        return self.df

    def setCorrelation(self, corr):
        self.corr = corr

    def setResultNode(self,node_name):
        self.result_node = node_name

    def setSampleSetSize(self,size):
        self.n = size

    def addNode(self,node_name,number_of_factors):
```

```
        max = self.df[node_name].max()
        min = self.df[node_name].min()
        eps = int((max-min)/number_of_factors)
        node_dict = {'min':min, 'max':max}


        for i in range(number_of_factors):
            filter = (self.df[node_name] <= int(min) + (i+1) * eps) & (self.df[node_name] > int(min) + i *
eps)
            node_dict["<={0}".format(int(min) + (i+1) * eps)] =
round(self.df[node_name][filter].count()/self.df[node_name].count(),2)

        self.nodes[node_name] = node_dict

    def addNode(self,node_name):
        node_dict = {}
        if node_name != "current_balance":
            for item in self.df[node_name]:
                if item in node_dict:
                    node_dict[item] += 1
                else:
                    node_dict[item] = 1

            for p in node_dict:
                node_dict[p] /= len(self.df)
        else:
            number_of_factors = 5
            max = self.df[node_name].astype(int).max()
            min = self.df[node_name].astype(int).min()
            eps = int(round((max - min) / number_of_factors))
            node_dict = {'min': min, 'max': max}

            for i in range(number_of_factors):
                filter = (self.df[node_name].astype(int) <= int(min) + (i + 1) * eps) &
(self.df[node_name].astype(int) > int(min) + i * eps)
                node_dict["<={0}".format(int(min) + (i + 1) * eps)] = round(
                    self.df[node_name][filter].count() / self.df[node_name].count(), 4)

            self.nodes[node_name] = node_dict

        self.nodes[node_name] = node_dict


    def getNodes(self):
        return self.nodes

    def getGraphDataFrame(self):
```

```
        self.graphDF = pd.DataFrame({'from': [], 'to': []})
        self.corr_matrix = self.corr_df
        self.buildNet([self.result_node])

        return self.graphDF

    def buildNet(self, start_columns):
        if start_columns.__len__() != 0:
            end_columns = []
            for column in start_columns:
                next_columns = self.corr_matrix[column][self.corr_matrix[column].index != column][
                    abs(self.corr_matrix[column]) > self.corr].keys()

                self.dependenses[column] = next_columns
                del self.corr_matrix[column]
                self.corr_matrix.drop(column, inplace=True)
                for column_node in next_columns:
                    if column_node not in start_columns:
                        self.graphDF = self.graphDF.append({'from': column_node, 'to': column},
ignore_index=True)
                        if column_node not in end_columns:
                            end_columns.append(column_node)
                            self.used_columns.append(column_node)

            self.buildNet(end_columns)

    def train(self):
        X = np.array(self.df[self.used_columns].values)
        Y = np.array(self.df['result'].values)


        self.model.fit(X,Y);


        prediction = self.model.predict(np.array(self.test_df[self.used_columns].values))

        print(prediction)
        print(self.test_df['result'].values)

        print(prediction-self.test_df['result'].values)

        return (self.n - np.count_nonzero(prediction-self.test_df['result'].values))



import matplotlib
from BayesianNetwork import *
matplotlib.use("TkAgg")
```

```
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg, NavigationToolbar2Tk
from matplotlib.figure import Figure
import networkx as nx
import matplotlib.pyplot as plt
import pandas as pd
from tkinter import filedialog as fd
import tkinter.scrolledtext as ScrolledText
import tkinter as tk

LARGE_FONT = ("Verdana", 12)


class BNmain(tk.Tk):

    def __init__(self, *args, **kwargs):
        tk.Tk.__init__(self, *args, **kwargs)

        tk.Tk.wm_title(self, "Byesian Network Diplom2019")
        container = tk.Frame(self)
        container.pack(side="top", fill="both", expand=True)
        container.grid_rowconfigure(0, weight=1)
        container.grid_columnconfigure(0, weight=1)
        self._filetypes =(
        ('Csv files', '*.csv'),
        ("All files", "*.*"))
        self._initaldir = r"C:\Users"

        self.filepath = tk.StringVar()
        self._create_widgets()
        self._display_widgets()
        self.slider = tk.Scale(self, from_=0, to=1, resolution=0.5, orient=tk.HORIZONTAL,
command=self.on_change_correlation)
        self.slider.pack()
        self.network = Network()




    def _create_widgets(self):
        self._entry = tk.Entry(self, textvariable=self.filepath)
        self._button = tk.Button(self, text="Browse...", command=self.browse)

    def _display_widgets(self):
        self._entry.pack(fill='x', expand=True)
        self._button.pack(anchor='se')

    def on_change_correlation(self, value):
```

```
        self.network.setCorrelation(value)
        self.G = nx.from_pandas_edgelist(self.network.getGraphDataFrame(), 'from', 'to')

    def browse(self):
        """ Browses a .png file or all files and then puts it on the entry.
        """

        file = fd.askopenfilename(initialdir=self._initaldir,
                                  filetypes=self._filetypes)
        self.filepath.set(file)
        self.network.setFileName(file)
        self.network.readDataFromFile()

        f = plt.figure(figsize=(8, 6))
        a = f.add_subplot(111)
        self.G = nx.from_pandas_edgelist(self.network.getGraphDataFrame(), 'from', 'to')
        pos = nx.circular_layout(self.G)
        nx.draw_networkx(self.G, pos=pos, ax=a)

        self.canvas = FigureCanvasTkAgg(f, master=self)
        self.canvas.draw()
        self.canvas.get_tk_widget().pack(side=tk.BOTTOM, fill=tk.BOTH, expand=True)

        tk.Button(self, text="Click for exit", command=self.destroy).pack()

        toolbar = NavigationToolbar2Tk(self.canvas, self)
        toolbar.update()
        self.canvas._tkcanvas.pack(side=tk.TOP, fill=tk.BOTH, expand=True)

        sText = ScrolledText.ScrolledText(self, height=20)
        sText.pack(side=tk.BOTTOM, fill=tk.BOTH, expand=True)

        for node in self.network.getNodes():
            sText.insert(tk.INSERT, str(node)+ '\n')
            sText.insert(tk.INSERT, str(self.network.getNodes()[node])+ '\n\n')


app = BNmain()
app.mainloop()
from BayesianNetwork import *
from View import View
import tkinter as tk
```

```
if __name__ == "__main__":
    app = View()
    app.mainloop()
```