Національний університет «Пролтавська політехніка імені Юрія Кондратюка»

Навчально науковий інститут інформаційних технологій та механотроніки

Кафедра комп’ютерних та інформаційних технологій і систем

**Пояснювальна записка**

**до дипломної роботи**

бакалавра

(освітньо-кваліфікаційний рівень)

на тему: «Розробка апаратно-програмних засобів оцінювання успішності здобувача вищої освіти з використанням нейромережевих технологій»

Виконав: студент 4 курсу, групи 402-ТК

спеціальності

\_\_123 Комп’ютерна інженерія \_

(шифр і назва напряму)

Кривенко В.В.\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Керівник Альошин С.П.\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Рецензент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Полтава – 2021 року

\_\_\_Національний університет «Полтавська політехніка імені Юрія Кондратюка»\_\_\_\_\_\_\_\_

(повне найменування вищого навчального закладу)

Факультет **\_**Навчально науковий інститут інформаційних технологій та механотроніки **\_**

Кафедра \_\_\_\_\_\_\_\_\_комп’ютерних та інформаційних технологій і систем\_\_\_\_\_\_\_\_\_\_\_

Освітньо-кваліфікаційний рівень \_\_\_\_\_\_\_\_бакалавр\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Напрям підготовки **\_\_**Комп’ютерна інженерія**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(шифр і назва)

Спеціальність **\_\_\_\_\_\_\_\_\_\_**123**\_\_\_** Комп’ютерна інженерія**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(шифр і назва)

**ЗАТВЕРДЖУЮ**

**Завідувач кафедри \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

"\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ року

**ЗАВДАННЯ**

**НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТУ**

|  |
| --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Кривенко Владислав Володимирович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (прізвище, ім’я, по батькові)  1. Тема проекту (роботи) Розробка апаратно-програмних засобів оцінювання успішності\_ здобувача вищої освіти з використанням нейромережевих технологій\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** керівник проекту (роботи) **\_**Альошин Сергій Павлович**,** доцент кафедри інформаційних технологій і систем\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**,                                                       (прізвище, ім’я, по батькові, науковий ступінь, вчене звання) затверджені наказом вищого навчального закладу від "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_ 2021 року № \_\_\_\_\_  2. Строк подання студентом проекту (роботи) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  3. Вихідні дані до проекту (роботи) Розробка апаратно-програмних засобів оцінювання успішності\_ здобувача вищої освіти з використанням нейромережевих технологій\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1) Вибір програмного забезпечення\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  2) Опис предметної області\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  3) Розробка нейромережі для оцінювання успішності здобувача вищої освіти\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  6. Консультанти розділів проекту (роботи) |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
| завдання видав | завдання прийняв |
| 1 | старший викладач Альошин С.П. | 07.04.2021 |  |
| 2 | старший викладач Альошин С.П. | 07.04.2021 |  |

|  |
| --- |
| 7. Дата видачі завдання \_\_\_\_\_\_\_\_\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  КАЛЕНДАРНИЙ ПЛАН |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| N з/п | | Назва етапів дипломного проекту (роботи) | | | Строк виконання етапів проекту (роботи) | Примітка |
| 1 | | Підготовчий (ознайомлення із завданнями, збирання фактичних матеріалів) | | | 17.04.2017 |  |
| 2 | | Основний (проведення консультацій, оформлення пояснювального матеріалу) | | | 22.05.2021 |  |
| 3 | | Заключний (отримування відзиву керівника, рецензії на дипломну роботу, проведення попереднього захисту на кафедрі) | | | 18.06.2021 |  |
| **Студент** | | **\_\_\_\_\_\_\_\_\_** (підпис) | \_\_\_\_\_\_\_\_\_Кривенко В.В\_\_\_\_\_\_\_\_\_  (прізвище та ініціали) | | | |
| **Керівник проекту (роботи)** | | **\_\_\_\_\_\_\_\_\_** (підпис) | Альошин С.П.\_\_\_\_\_\_\_\_\_  (прізвище та ініціали) | | | |

# РЕФЕРАТ

# Загальний обсяг текстової частини дипломної роботи 66 сторінок формату А4. Робота складається з переліку скорочень, умовних позначень; вступу; трьох розділів; висновків; списку використаних джерел;. Вона містить 30 рисунків, 3 таблиці.

# Об’єкт дослідження – процес розробки нейромережі для знаходження підсумкового середнього балу для студентів, а також спрогнозувати успішність навчання.

# Предмет дослідження – розглянути основні теоретичні аспекти нейромережевого програмного продукту StatSoft STATISTICA 6.1, а також основні методи розробки нейромережі.

# Наукове та практичне значення роботи: оновлення способів оцінювання успішності студентів за допомогою штучного інтелекту.

# 

# ЗМІСТ

[ВСТУП 6](#_Toc436342965)

[1. ТЕОРЕТИЧНІ ВІДОМОСТІ 9](#_Toc436342966)

## 1.1 Продукційна модель………………………………………………………………….. 9

1.2 Мережева модель…………………………………………………………………...15

1.3 Фреймова модель…………………………………………………………………...18

1.4 Логічна модель………………………………………………………………………25

[2.ПОСТАНОВКА ЗАДАЧІ 27](#_Toc436342967)

[3. ПРАКТИЧНА ЧАСТИНА 30](#_Toc436342968)

[Висновок 49](#_Toc436342969)

[Список використаних джерел 50](#_Toc436342970)

[Додаток A 52](#_Toc436342970)

# 

# ВСТУП

Мета даної роботи, використовуючи процес навчання, що є процедурою реалізації штучного інтелекту, створити нейромережу, яка буде з мінімальною помилкою і найбільшою продуктивністю, використовувати процес навчання, що є процедурою реалізації штучного інтелекту. Практично вирішувати задачу знаходження підсумкового середнього балу для студентів, а також спрогнозувати успішність навчання.

Нейромережа – це сукупність нейронів з’єднаних одне з одним певним чином. Діяльність мозку, спрямовану на вирішення інтелектуальних завдань, будемо називати мисленням, або інтелектуальною діяльністю. Інтелект і мислення органічно пов'язані з вирішенням таких завдань, як доводження теорем, логічний аналіз, розпізнавання ситуацій, планування поводження, ігри і керування в умовах невизначеності. Характерними рисами інтелекту, що проявляються в процесі вирішення завдань, є здатність до навчання, узагальнення, нагромадженню досвіду (знань і навичок) та адаптації до мінливих умов в процесі вирішення завдань. Завдяки цим якостям інтелекту мозок може вирішувати різноманітні завдання, а також легко перебудовуватися з рішення однієї задачі на іншу. Таким чином, мозок, наділений інтелектом, є універсальним засобом вирішення широкого кола завдань (у тому числі неформалізованих) для яких немає стандартних, заздалегідь відомих методів вирішення. І тому увагу вчених до логічного моделювання процесів головного мозку викликали такі умови:

* висока швидкість виконання важких логічних конструкцій;
* простота алгоритмів логічних дій мозку, заснована не на числовому маніпулюванні, а на принципах асоціативного мисленням;
* можливість рішення важко формалізованих задач, в яких сумісно використовуються дані логічно несумісної природи;
* можливість побудови самонавчающих систем;

Іншим прикладом поведінкового трактування інтелекту може служити відоме визначення А. Тьюрінга. Його зміст полягає в наступному. У різних кімнатах знаходиться люди і машина. Вони не можуть бачити один одного, але мають можливість обмінюватися інформацією (наприклад, за допомогою електронної пошти). Якщо в процесі діалогу між учасниками гри людям не вдається встановити, що один з учасників - машина, то така машина можна вважати володіє інтелектом.

До речі цікавий план імітації мислення, запропонований А. Тьюрінгом. “Намагаючись імітувати інтелект дорослої людини, - пише Тьюрінг, - ми змушені багато міркувати про той процес, в результаті якого людський мозок досяг свого дійсного стану. Чому б нам замість того, щоб намагатися створити програму, що імітує інтелект дорослої людину, не спробувати створити програму, яка імітувала б інтелект дитини? Адже якщо інтелект дитини отримує відповідне виховання, він стає інтелектом дорослої людини Наш розрахунок полягає в тому, що пристрій, і йому подібне, може бути легко запрограмоване.”

Ряд етапів рішення задачі методами ІАД:

1. Постановка завдання аналізу;
2. Збір даних;
3. Підготовка даних (фільтрація, доповнення, кодування);
4. Вибір моделі (алгоритму аналізу даних);
5. Підбір параметрів моделі та алгоритму навчання;
6. Навчання моделі (автоматичний пошук решти параметрів моделі);
7. Аналіз якості навчання, якщо незадовільний перехід на п. 5 або п. 4;
8. Аналіз виявлених закономірностей, якщо незадовільний перехід на п. 1, 4 або 5.

База знань - важливий компонент експертної системи, вона призначена для зберігання довгострокових даних, що описують розглянуту предметну область (а не поточних даних), і правил, що описують доцільні перетворення даних цієї області.

Далі для створення ЕС в обраній області збираються факти і правила, що містяться в базі знань разом з механізмами виведення і спрощення. На відміну від всіх інших компонентів ЕС, база знань - "змінна" частина системи, яка може поповнюватися і модифікуватися інженерами знань і досвіду використання ЕС, між консультаціями (а в деяких системах і в процесі консультації).

Існує кілька способів подання знань в ЕС, однак загальним для всіх них є те, що знання представлені в символьній формі (елементарними компонентами представлення знань є тексти, списки та інші символьні структури). Тим самим, в ЕС реалізується принцип символьної природи міркувань, який полягає в тому, що процес міркування представляється як послідовність символьних перетворень.

Існують динамічні і статичні бази знань. Динамічна база знань змінюється з часом. Її вміст залежить і від стану навколишнього. Нові факти, що додаються в базу знань, є результатом виведення, який складається в застосуванні правил до наявних фактів. У системах з монотонним висновком факти, збережені в базі знань, статичні, тобто не змінюються в процесі виконання завдання. У системах з немонотонним висновком допускається зміна або видалення фактів з бази знань.

Однією з найбільш важливих проблем, характерних для систем, заснованих на знаннях, є проблема подання знань. Це пояснюється тим, що форма подання знань має суттєвий вплив на характеристики і властивості системи. Для того щоб маніпулювати всілякими знаннями з реального світу за допомогою комп'ютера, необхідно здійснювати їх моделювання. У таких випадках необхідно відрізняти знання, призначені для обробки комп'ютером, від знань, які використовуються людиною.

При проектуванні моделі подання знань слід враховувати такі фактори, як однорідність подання і простота розуміння. Однорідне уявлення призводить до спрощення механізму керування логічним висновком і спрощення управління знаннями. Подання знань має бути зрозумілим експертам і користувачам системи. В іншому випадку можуть бути придбання знань і їх оцінка. Однак виконати цю вимогу в рівній мірі, як для простих, так і для складних завдань досить важко. Зазвичай, для нескладних завдань зупиняються на деякому середньому (компромісному) поданні, але для вирішення складних і великих завдань необхідні структурування і модульне подання.

На сьогоднішній день розроблено вже достатня кількість моделей представлення знань. Кожна з них має свої плюси і мінуси, і тому для кожного окремого завдання необхідно вибрати саме свою модель. Від цього буде залежить не стільки ефективність виконання поставленого завдання, скільки можливість його рішення взагалі.

Моделі подання знань відносяться до прагматичного напрямку досліджень в області штучного інтелекту. Цей напрямок заснований на припущенні про те, що розумова діяльність людини - «чорний ящик». При такому підході не ставиться питання про адекватність використовуваних в комп'ютері моделей уявлення знань тим моделям, якими користується в аналогічних ситуаціях людина, а розглядається лише кінцевий результат вирішення конкретних завдань.

**1.МОДЕЛІ ПОДАННЯ ЗНАНЬ**

1.1. Продукційна модель

Продукційні моделі можна вважати найбільш поширеними моделями подання знань. Продукційна модель - модель, заснована на правилах, що дозволяє уявити знання у вигляді пропозицій типу:

«ЯКЩО умова, ТО дія»

Психологічні дослідження процесів прийняття рішень людиною показали, що розмірковуючи і приймаючи рішення, людина використовує правила продукцій, або продукційні правила (від англ. Production - правило виводу, що породжує правило).

У загальному випадку продукційне правило можна представити в наступному вигляді:

У загальному випадку під умовою розуміється деякя пропозиція - зразок, за яким здійснюють пошук в базі знань, а під дією - дії, що виконуються при успішному результаті пошуку, - це можуть бути реальні дії, якщо система керуюча, або висновок, що представляє собою нове знання, або деяку мету.

При використанні продукційної моделі база знань складається з набору правил. Програма, що управляє перебором правил, називається машиною виведення. Механізм висновків пов'язує знання воєдино, а потім виводить з послідовності знань висновок.

В якості умов і дій в правилах може бути, наприклад, припущення про наявність тієї чи іншої властивості, що приймає значення істина або брехня. При цьому термін «дій» слід трактувати широко: це може бути як директива до виконання будь-якої операції, рекомендація, або модифікація бази знань - припущення про наявність будь-якого похідного властивості.

При використанні продукційної моделі база знань складається з набору правил. Програма, що управляє перебором правил, називається машиною виведення. Найчастіше висновок буває прямий (від даних до пошуку мети) або зворотний (від мети для її підтвердження - до даних). Дані - це вихідні факти, на підставі яких запускається машина виведення - програма, перебирає правила з бази.

У загальному випадку продукційну модель можна представити в наступному вигляді:

i: S; L; A → B; Q, (1.1)

де, i - індивідуальний номер продукції;

S - опис класу ситуацій, в якому дана структура може використовуватися;

L - умова, при якому продукція активізується;

А → В - ядро ​​продукції, наприклад: «ЯКЩО A1, A2 , ..., Аn ТО В». Такий запис означає, що «якщо всі умови від A1 до Аn є істиною, то В також істина» або ж «коли всі умови від A1 до Аn стають істиною, то слід виконати дію B»;

Q - підумова продукційного правила, описує операції і дії (процедури), які необхідно виконати після виконання В. Наприклад, внести зміни в дані або в саму продукцію.

Приклад:

ЯКЩО

у є батьком х, (A 1)

z є братом у, (A 2)

ТО

z є дядьком х, (В)

В цьому випадку n = 2. При n = 0 отримуємо знання, які складаються тільки з виведення, тобто простий факт, наприклад, «Атомна вага заліза дорівнює 55,8471». Суть використання правил продукції для представлення знань полягає в тому, що лівій частині ставиться у відповідність деяка умова, а правій частині - дія: ЯКЩО <перелік умови>, ТО <перелік дій>. У такій інтерпретації ліва частина правил оцінюється по відношенню до бази даних (відомому набору фактів) системи, і якщо ця оцінка в певному сенсі відповідає логічному значенню «ІСТИНА», то виконується дія, задана в правій частині продукції.

Приклад:

Нехай в базі знань разом з описаними вище знаннями містяться ще й такі знання:

ЯКЩО z є батьком х,

z є батьком у,

х і у не є одним і тим же особою,

ТО х і у є братами;

де х, у, z - змінні.

Нехай також в базі даних (робочої пам'яті) є факти:

α є батьком β,

α є батьком γ,

β є батьком δ.

Тоді з цих знань можна формально вивести висновок, що

γ є дядьком δ.

У продукційних системах, заснованих на знаннях, процес обробки інформації може здійснюватися двома способами. Перший передбачає обробку інформації в прямому напрямку (метод зіставлення), коли зразком для пошуку служить ліва частина продукційного правила - умова, тобто завдання вирішується в напрямку від початкового стану до цільового. Це відповідає стратегії «від даних до мети» або стратегії управління даними. Після вирішення конфліктів виконуються праві частини продукційних правил, що відповідають логічному висновку нових тверджень. Після додавання виведених тверджень в базу даних процедура повторюється. Процес закінчується, якщо виконується продукційне правило, що наказує припинення пошуку, або в базу даних надходить твердження, що є рішенням. При іншому підході обробка інформації здійснюється в зворотному напрямку - метод «генерації» або висунення гіпотези і її перевірки (стратегія «від мети до даних»). При кожному зворотному русі виникає під цільовий стан, з якого цільове може бути отримано при прямому русі. В цьому випадку перевіряються праві частини продукційних правил з метою виявлення в них шуканого твердження. Якщо такі продукційні правила існують, то перевіряється, чи ліва частина продукційного правила задовольняє умову. Якщо так, то гіпотеза вважається підтвердженою, якщо не підтверджується - відкидається.

Таким чином, продукційні правила можуть застосовуватися до опису стану програми і описувати нові стани (гіпотези) або ж, навпаки, використовувати цільовий стан завдання як базу, коли система працює в зворотному напряму. При цьому продукційні правила застосовуються до цільового опису для породження ряду цілей (утворюють систему редукцій).

Приклад:

Є фрагмент БЗ з двох правил:

П 1: ЯКЩО «відпочинок - влітку» і «людина - активний»,

ТО «їхати в гори».

П 2: ЯКЩО «любить сонце», «Відпочинок влітку».

Припустимо в систему надійшли дані:

«Людина - активний» і «любить сонце»

Прямий висновок: виходячи з даних, отримати відповідь.

1-й прохід:

Крок 1. Пробуємо П1 не працює -

не вистачає даних «відпочинок - влітку».

Крок 2. Пробуємо П2, працює,

в базу надходить факт «відпочинок - влітку». 2-й прохід:

Крок 3. Пробуємо П1, працює,

активуючи мета «їхати в гори», яка і виступає, наприклад, як порада, яку дає система.

Зворотний висновок:

підтвердити обрану мету за допомогою наявних правил і даних. 1-й прохід:

Крок 1. Мета - «їхати в гори»:

стають новою метою, і є правило, де вона в правій частині.

Крок 2. Мета «відпочинок влітку»:

правило П2 підтверджує мету і активізує її. 2-й прохід:

Крок 3. Пробуємо П1, підтверджується шукана мета.

Системи обробки знань, що використовують продукційну модель отримали назву «продукційні системи». До складу експертних систем продукційного типу належать база правил (знань), робоча пам'ять і інтерпретатор правил (вирішувач), який реалізує певний механізм логічного висновку. Будь-яке продукційне правило, яке міститься в базі знань, складається з двох частин: антецендент і консеквента. Антецедент представляє собою посилку правила (умовну частину) і складається з елементарних пропозицій, з'єднаних логічними зв'язками «і», «або». Консеквент (висновок) включає одне або кілька речень, які висловлюють або деякий факт, або вказівку на певну дію, яка підлягає виконанню. Продукційні правила прийнято записувати у вигляді антецедент-консеквент.

Властивості продукційних моделей:

Модульність - окремі продукційні правила можуть бути додані, видалені або змінені в базі знань незалежно від інших; крім того, модульний принцип розробки (складання) продукційних систем дозволяє автоматизувати їх проектування.

Кожне продукційне правило - самостійний елемент знань (локальне джерело знань); окремі продукційні правила пов'язані між собою тільки через потік даних, які вони обробляють.

Простота інтерпретації - «прозора» структура продукційних правил полегшує їх смислову інтерпретацію.

Природність - знання у вигляді «що робити і коли» є природніми з точки зору здорового глузду.

Продукційні моделі (поряд з фреймами) є найбільш поширеними застосовуваними засобами подання знань. Вони близькі до логічних моделей, що дозволяє організовувати на їх базі ефективні процедури виведення, і в той же час більш наочно (порівняно з класичними логічними моделями) відображають знання. Продукційна модель найчастіше застосовується в промислових експертних системах. Вона привертає розробників своєю наочністю, високою модульністю, легкістю внесення доповнень і змін і простотою логічного висновку.

Основні переваги систем, заснованих на продукційних моделях, пов'язані з простотою подання знань і організації логічного висновку.

Недоліки продукційних систем проявляються тоді, коли число правил стає великим і виникають непередбачувані побічні ефекти від змін старих і додання нових правил. Крім того, є важким оцінка цілісного образу знань, що містяться в системі. До інших недоліків відносяться:

* відміна від структур знань, властивих людині;
* неясність взаємних відносин правил;
* складність оцінки цілісного образу знань;
* низька ефективність обробки знань.

При розробці невеликих систем (десятки правил) проявляються в основному позитивні сторони продукційних моделей знань, проте при збільшенні обсягу знань більш помітними стають слабкі сторони.

1.2. Мережева модель

Семантична модель або мережева модель – це спрямований граф, що відображає зміст цілісного образу з пойменованими вершинами і дугами. Вузли графа відповідають поняттям і об'єктам, а дуги - відносинам між об'єктами;

В якості понять зазвичай виступають абстрактні або конкретні об'єкти, події, властивості, дії.

Дана модель подання знань була запропонована американським психологом Куілліаном.

В основі моделей цього типу лежить конструкція, названа семантичною мережею. Семантичний підхід до побудови систем штучного інтелекту знаходить застосування в системах розуміння природної мови, в питально-відповідних системах, в різних предметно-орієнтованих системах.

Термін семантична означає змістовність, а сама семантика - це наука, що встановлює відносини між символами і об'єктами, які вони позначають, тобто наука, яка визначає зміст знаків.

У найзагальнішому випадку семантична мережа являє собою інформаційну модель предметної області і має вигляд грифа, вершини якого відповідають об'єктам предметної області, а дуги - відносинам між ними.

Дуги можуть бути визначені різними методами, які залежать від виду знань, що подаються. Зазвичай дуги, які використовуються для подання ієрархії, включають дуги типу «безліч», «підмножина», «елемент». Семантичні мережі, що застосовуються для опису природних мов, використовують дуги типу «агент», «об'єкт», «реципієнт».

Поняттями зазвичай виступають абстрактні або конкретні об'єкти, а відносинами - зв'язки типу: «це» ( «is»), «має частиною» ( «has part»), «належить», «любить». Характерною особливістю семантичних мереж є обов'язкова наявність трьох типів відносин:

* клас - елемент класу (квітка – тюльпан);
* властивість – значення ( квітка - жовта);
* приклад елемента класу ( тюльпан шренка).

Можна ввести декілька класифікацій семантичних мереж. Наприклад, за кількістю типів відносин:

* однорідні (з єдиним типом відносин);
* неоднорідні (з різними типами відносин).

За типами відносин:

* бінарні (в яких відносини пов'язують два об'єкти);
* парні (в яких є спеціальні відносини, що зв'язують більше двох понять).

Найбільш часто в семантичних мережах використовуються наступні відносини:

* зв'язку типу «частина-ціле» ( «клас-підклас», «елемент-множина» і т.п.);
* функціональні зв'язки (визначені зазвичай дієсловами «виробляє», «впливає» ...);
* кількісні (більше, менше, дорівнює ...);
* просторові (далеко від, близько від, за, під, над ...);
* тимчасові (раніше, пізніше, протягом ...);
* атрибутивні зв'язку (мати властивість, мати значення ...);
* логічні зв'язки (і, або, не) і ін.

Приклад 1. «Queen Mary є океанським лайнером» і «Кожен океанський лайнер є кораблем». Вони можуть бути представлені через семантичну мережу.

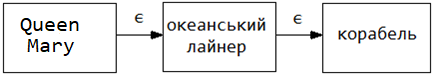


Рисунок 1.2.1 - Приклад семантичної мережі

Приклад 2. Поняття «корабель».



Рисунок. 1.2.2 - Семантична мережа для поняття «корабель»

Проблема пошуку рішення в базі знань типу семантичної мережі зводиться до задачі пошуку фрагмента мережі, відповідно деякої підмережі, що відповідає поставленому питанню.

У семантичних мережах існує можливість представляти знання більш природним і структурованим чином, ніж в інших формах.

Основною перевагою є те, що вона більше за інших відповідає сучасних уявленням про організацію довготривалої пам'яті людини.

Недоліком цієї моделі є складність організації процедури пошуку виведення на семантичній мережі.

Для реалізації семантичних мереж існують спеціальні мережеві мови (NET, SIMER + MIR та ін.). Відомі системи, що використовують семантичні мережі в якості мови уявлення знань-PROSPECTOR, CASNET, TORUS.

1.3. Фреймова модель

Фреймова модель представлення знань - була запропонована М. Мінська у 1979 році як структура знань для сприйняття просторових сцен. Ця модель, як і семантична мережа, має глибоке психологічне обґрунтування.

У психології та філософії відоме поняття абстрактного образу. Наприклад, слово «кімната» викликає у слухачів образ кімнати: «житлове приміщення з чотирма стінами, підлогою, стелею, вікнами та дверима, площею 6-20 м2». З цього опису нічого не можна прибрати (наприклад, прибравши вікна, отримаємо вже комору, а не кімнату), але в ньому є «дірки», або «слоти», - це незаповнені значення деяких атрибутів - кількість вікон, колір стін, висота стелі, покриття підлоги й ін. У цій теорії такий абстрактний образ називається фреймом.

Фреймом називається також і формалізована модель для відображення образу.

В якості ідентифікатора кадру присвоюється ім'я фрейма. Це ім'я має бути єдиним у всій фреймовой системі.

Фрейм має певну внутрішню структуру, що складається з безлічі елементів, які називаються слотами, яким також присвоюються імена. За слотами слідують шпації, в які поміщають дані, що представляють поточні значення слотів. Кожен слот в свою чергу представляється певною структурою даних. В значення слота підставляється конкретна інформація, що відноситься до об'єкту, що описується цим фреймом.

Структуру фрейма можна представити у наступному вигляді:

Ім'я фрейма:

(Ім'я 1-го слота: значення 1-го слота),

(Ім'я 2-го слота: значення 2-го слота),

- - - -

(Ім'я N-го слота: значення N-гo слота).

Цей же запис представлено у вигляді таблиці, у якій додано ще два стовпця.

Таблиця 1.3.1

|  |  |  |  |
| --- | --- | --- | --- |
| Ім’я слота | Значення слота | Спосіб отримання слота | Приєднана процедура |
|  |  |  |  |
|  |  |  |  |

У таблиці додаткові стовпці призначені для опису способу отримання слотом його значення і можливого приєднання до того чи іншого слоту спеціальних процедур, що допускається в теорії фреймів. Як значення слота може виступати ім'я іншого фрейма; так утворюються мережі фреймів.

Розрізняють фрейми-зразки, або прототипи, що зберігаються в базі знань, і фрейми-екземпляри, які створюються для відображення реальних ситуацій на основі даних, що надходять.

Модель фрейма є досить універсальною, оскільки дозволяє відобразити все різноманіття знань про світ:

* через фрейми-структури, для позначення об'єктів і понять (позика, застава, вексель);
* через фрейми-ролі (менеджер, касир, клієнт);
* через фрейми-сценарії (банкрутство, збори акціонерів, святкування іменин);
* через фрейми-ситуації (тривога, аварія, робочий режим пристрою) і тд.

Найважливішою властивістю теорії фреймів є запозичене з теорії семантичних мереж успадкування властивостей. І у фреймах, і в семантичних мережах спадкування відбувається за AKO-зв'язків (A-Kind-Of = це). Слот АКО вказує на фрейм більш високого рівня ієрархії, звідки неявно успадковуються, тобто переносяться, значення аналогічних слотів.

Значним слота може бути практично що завгодно: числа, формули, тексти на природній мові або програми, правила виведення або посилання на інші слоти даного фрейму або інших фреймів. Як значення слота може виступати набір слотів більш низького рівня, що дозволяє реалізовувати у фреймових уявленнях «принцип матрьошки». Зв'язки між фреймами задаються значеннями спеціального слота з ім'ям «Зв'язок».

У загальному випадку структура даних фрейму може містити більш широкий набір інформації, до якого входять такі атрибути.

**Ім'я фрейму.** Воно служить для ідентифікації фрейма в системі і має бути унікальним. Фрейм представляє собою сукупність слотів, число яких може бути довільним. Число слотів в кожному фреймі встановлюється проектувальником системи, при цьому частина слотів визначається самою системою для виконання специфічних функцій (системні слоти), прикладами яких є: слот-покажчик батька даного фрейма (IS-А), слот-покажчик дочірніх фреймів, слот для введення імені користувача, слот для введення дати визначення фрейму, слот для введення дати зміни фрейма і т.д.

**Ім'я слота.** Воно повинно бути унікальним в межах фрейму. Зазвичай ім'я слота є ідентифікатор, який наділений певною семантикою. Як ім'я слота може виступати довільний текст. Наприклад, <Ім'я слота> = Головний герой роману Ф.М. Достоєвського «Ідіот», «Значення слота> = Князь Мишкін. Імена системних слотів зазвичай зарезервовані, в різних системах вони можуть мати різні значення. Приклади імен системних слотів: IS-A, HASPART, RELATIONS і т.д. Системні слоти служать для редагування бази знань і управління виводу у фреймовій системі.

**Показники наслідування.** Вони показують, яку інформацію про атрибути слотів з фрейма верхнього рівня успадковують слоти з аналогічними назвами в даному фреймі. Покажчики успадкування характерні для фреймових систем ієрархічного типу, заснованих на відносинах типу «абстрактне - конкретне». У конкретних системах покажчики успадкування можуть бути організовані різними способами і мати різні позначення:

U (Unique) - значення слота не успадковується;

S (Same) - значення слота успадковується;

R (Range) - значення слота повинні знаходитися в межах інтервалу значень, зазначених в однойменному слоті батьківського фрейма;

O (Override) - при відсутності значення в поточному слоті воно успадковується з фрейма верхнього рівня, однак в разі визначення значення поточного слота воно може бути унікальним. Цей тип покажчика виконує одночасно функції покажчиків U і S.

**Показник типу даних**. Він показує тип значення слота. Найбільш вживані типи: frame - покажчик на фрейм; real - дійсне число; integer - ціле число; boolean - логічний тип; text - фрагмент тексту; list - список; table - таблиця; expression - вираз; lisp - пов'язана процедура і т.д.

**Значення слота.** Воно повинно відповідати зазначеному типу даних і умові успадкування.

**Демони.** Демоном називається процедура, що автоматично запускається при виконанні деяких умов. Демони автоматично запускаються при зверненні до відповідного слоту. Типи демонів пов'язані з умовою запуску процедури. Демон з умовою IF-NEEDED запускається, якщо в момент звернення до слоту його значення не було встановлено. Демон типу IF-ADDED запускається при спробі зміни значення слота. Демон IF-REMOVED запускається при спробі видалення значення слота. Можливі також інші типи демонів. Демон є різновидом пов'язаної процедури.

**Приєднана процедура.** Як значення слота може використовуватися процедура, яка називається службовою в мові Лісп або методом в мовах об'єктно-орієнтованого програмування. Приєднана процедура запускається за повідомленням, яке було передано з іншого фрейму. Демони і приєднані процедури є процедурними знаннями, об'єднаними разом з декларативними в єдину систему. Ці процедурні знання є засобами управління виводу у фреймових системах, причому з їх допомогою можна реалізувати будь-який механізм виведення. Подання таких знань і заповнення ними інтелектуальних систем - дуже нелегка справа, яка вимагає додаткових витрат праці та часу розробників. Тому проектування фреймових систем виконується, як правило, фахівцями, що мають високий рівень кваліфікації в галузі штучного інтелекту.

Частина фахівців із систем штучного інтелекту вважають, що немає необхідності виділення фреймових моделей подання знань, так як в них об'єднані всі основні особливості моделей інших типів.

Нижче наведено декілька прикладів фреймових описів.

Приклад 1. Фрейм, що описує людину.

|  |  |
| --- | --- |
| **Фрейм:** | **Людина** |
| Ім’я слота: | Значення слота |
| Клас: | Тварина |
| Структурний елемент: | Голова, шия, руки, ... |
| Ріст: | 30 ? 220 см |
| Маса: | 1 ? 200 кг |
| Хвіст: | Ні |
| Мова: | Російська, англійська, ... |
| Зв’язок: | Мавпа |

Приклад 2. Опис з допомогою фрейму поняття письмового звіту.

У вигляді семантичної мережі «звіт» можна подати в наступному вигляді.

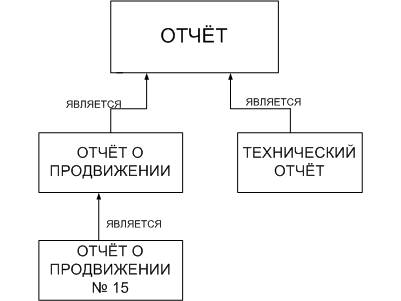


Рисунок 1.3.1 - Семантична мережа поняття «звіт»

Нижче поняття «звіт» подано з допомогою фреймів.

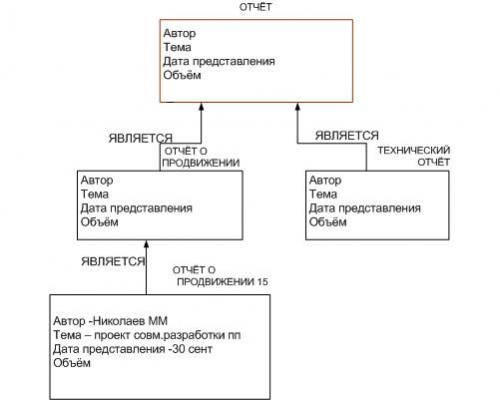


Рисунок 1.3.2 - Подання поняття «звіт» у вигляді фреймів

Основною перевагою фреймів як моделі подання знань є те, що вони відображають концептуальну основу організації пам'яті людини, а також її гнучкість і показовість. Найбільш яскраво переваги фреймових систем подання знань проявляються в тому випадку, коли родинні зв'язки змінюються нечасто і предметна область нараховує трохи винятків. У фреймових системах дані про родинні зв'язки зберігаються явно, як і знання інших типів. Значення слотів представляються в системі в єдиному екземплярі, оскільки включаються тільки в один фрейм, і описує найбільш зрозумілі поняття з усіх тих, які містить слот з такою назвою. Така властивість систем фреймів забезпечує економне розміщення бази знань в пам'яті комп'ютера. Ще одна перевага фреймів полягає в тому, що значення будь-якого слота може бути обчислено за допомогою відповідних процедур або знайдено евристичними методами. Тобто фрейми дозволяють маніпулювати як декларативними, так і процедурними знаннями.

До недоліків фреймових систем відносять їх відносно високу складність, що проявляється в зниженні швидкості роботи механізму виведення і збільшення трудомісткості внесення змін до родової ієрархію. Тому при розробці фреймових систем приділяють науковим способам відображення і ефективних засобів редагування фреймових структур.

Спеціальні мови представлення знань в мережах фреймів FRL (Frame Representation Language), KRL (Knowledge Representation Language), фреймова оболонка Kappa, PILOT / 2 і інші програмні засоби дозволяють ефективно будувати промислові системи.

В останні роки термін «фреймовий» часто замінюють терміном «об'єктно-орієнтований». Цей підхід є розвитком фреймового уявлення. Шаблон фрейма можна розглядати як клас, екземпляр фрейма - як об'єкт. Мови об'єктно-орієнтованого програмування (ООП) надають засоби створення класів і об'єктів, а також кошти для опису процедур обробки об'єктів (методи). Мови ООП, що не містять засобів реалізації приєднаних процедур, не дозволяють організувати гнучкий механізм логічного висновку, тому розроблені на них програми або являють собою об'єктно-орієнтовані бази даних, або вимагають інтеграції з іншими засобами обробки знань (наприклад, з мовою PROLOG).

Об'єктно-орієнтована методологія представлення знань реалізована в системах G2, RTWorks.

1.4. Логічна модель

В основі логічних моделей представлення знань лежить поняття формальної теорії, що задається формулою:

*М* = <*Т*, *Р*, *А*, *F*>, (1.4)

Де, *Т* — множина базових елементів;

*Р* — множина синтаксичних правил, на основі яких з *Т* будуються правильно побудовані формули;

*А* — множина правильно побудованих формул (аксіом);

*F* — правила виводу, які з безлічі *А* дозволяють отримувати нові правильно побудовані формули — теореми.

Найбільше поширення при побудові логічних моделей отримало числення предикатів 1-го порядку, що дозволяють не тільки забезпечити представлення знань для деякої ПрГ, але і здійснювати ефективний висновок при використанні методу резолюцій. Розробка методу резолюцій була основою створення мови логічного програмування *PROLOG*, який досить ефективно використовується при формалізації знань в самих різноманітних ПрГ.

Прикладом запису факту “3-му *мп* пройти вихідний рубіж у 6.00” може служити така формула обчислення предикатів:

ПРОЙТИ (3 мп, ВИХІДНИЙ РУБІЖ, 6.00).

Переваги логічної моделі представлення знань:

* Як «фундамент» тут використовується класичний апарат математичної логіки, методи якої досить добре вивчені і формально обґрунтовані.
* Існують досить ефективні процедури виведення, в тому числі реалізовані в мові логічного програмування Пролог, що використовують механізми автоматичного доведення теорем для пошуку і логічно осмисленого виведення інформації.
* У базах знань можна зберігати лише безліч аксіом, а всі інші знання отримувати з них за правилами виведення, а також Дані, факти та інші відомості про людей, предмети, події та процеси

# 2. ПОСТАНОВКА ЗАДАЧІ

Під час виконання роботи необхідно розробити нейромережу, яка буде з мінімальною помилкою і найбільшою продуктивністю, використовуючи процес навчання, що є процедурою реалізації штучного інтелекту, вирішувати задачу знаходження підсумкового середнього балу для студентів, а також спрогнозувати значення змінної на основі вхідної змінної.

Маємо задачу для визначення середнього бала для кожного студента в системі ECTS, що вираховується згідно оцінок з 10-ти дисциплін. За допомогою цих даних необхідно визначити середній бал для інших студентів. Прогнозування буде проводитися на основі задачі класифікації із застосуванням пакета технічного аналізу SТАTISTIСA 6.1

Найважливішою властивістю нейронних мереж є їх здатність навчатися на основі даних навколишнього середовища і в результаті навчання підвищувати свою продуктивність. Підвищення продуктивності відбувається з часом у відповідності з певними правилами. Навчання нейронної мережі відбувається за допомогою інтерактивного процесу коригування синаптичних ваг і порогів. В ідеальному випадку нейронна мережа отримує знання про довкілля на кожній ітерації процесу навчання.

З поняттям навчання асоціюється досить багато видів діяльності, тому складно надати цьому процесу однозначне визначення. Більше того, процес навчання залежить від точки погляду на нього. Саме це робить практично неможливою появу якого-небудь точного визначення цього поняття. Наприклад, процес навчання з погляду психолога в корені відрізняється від навчання з погляду шкільного вчителя. З позицій нейронної мережі, ймовірно, можна використовувати наступне визначення:

Навчання - це процес, в якому вільні параметри нейронної мережі налаштовуються за допомогою моделювання середовища, в яке ця мережа була або буде влаштована; тип навчання визначається способом підлаштування цих параметрів.

Це визначення процесу навчання нейронної мережі припускає наступну послідовність подій:

1. У нейронну мережу надходять стимули із зовнішнього середовища.
2. В результаті першого пункту змінюються вільні параметри нейронної мережі.
3. Після зміни внутрішньої структури нейронна мережа відповідає на збудження вже іншим чином.

Вищевказаний список чітких правил вирішення проблеми навчання нейронної мережі називається алгоритмом навчання. Нескладно здогадатися, що не існує універсального алгоритму навчання, відповідного для всіх архітектур нейронних мереж. Існує лише набір компонентів і засобів, представлений безліччю алгоритмів навчання, кожен з яких має свої переваги. Алгоритми навчання відрізняються один від одного способом налаштування синаптичних ваг нейронів. Ще однією відмінною характеристикою є спосіб зв'язку навченої нейронної мережі із зовнішнім світом. У цьому контексті говорять про парадигму навчання, пов'язану з моделлю навколишнього середовища, в якому функціонує дана нейронна мережа.

Існують два концептуальних підходи до навчання нейронних мереж: навчання з учителем і навчання без учителя.

Навчання нейронної мережі з учителем припускає, що для кожного вхідного вектора з навчальної множини існує необхідне значення вихідного вектора, званого цільовим. Ці вектори утворюють навчальну пару. Ваги мережі змінюють до тих пір, поки для кожного вхідного вектора не буде отриманий прийнятний рівень відхилення вихідного вектора від цільового.

Навчання нейронної мережі без вчителя є набагато більш правдоподібною моделлю навчання з погляду біологічних коренів штучних нейронних мереж. Навчальна множина складається лише з вхідних векторів. Алгоритм навчання нейронної мережі підлаштовує ваги мережі так, щоб виходили узгоджені вихідні вектори, тобто щоб пред'явлення досить близьких вхідних векторів давало однакові виходи.

Перед використанням нейронної мережі її необхідно навчити. Процес навчання нейронної мережі полягає в підлаштуванні внутрішніх параметрів відповідно до вирішуваного завдання. Під ваговим коефіцієнтом розуміють числовий відносний показник, який характеризує ступінь складності (вагомості або важливості) певного тестового завдання. Довірчими порогами називають критерії проходження рішення через нейрон.

Алгоритм роботи нейронної мережі є ітеративним, кроки якого називають епохами або циклами. Епоха – одна ітерація в процесі навчання, на кожній ітерації в нейронну мережу подається уся навчальна вибірка (по одному спостереженню за раз) та можливість перевірки якості навчання на контрольній множині.

Крос-ентропія — це функція помилок, заснована на теоретико-інформаційних характеристиках. Особливо добре підходить для задач класифікації.

Розглянемо ієрархічну мережеву структуру, в якій пов'язані між собою нейрони (вузли мережі) об'єднані в кілька шарів. Можливість побудови таких архітектур вказав ще Ф.Розенблатт, проте він не зміг вирішити проблему навчання. Міжнейронні синаптичні зв'язки мережі влаштовані таким чином, що кожен нейрон на даному рівні ієрархії приймає і обробляє сигнали від кожного нейрона більш низького рівня. Таким чином, в даній мережі є виділений напрям розповсюдження нейроімпульсів - від вхідного прошарку через один (або декілька) прихованих шарів до вихідного прошарку нейронів. Нейромережу такої топології будемо називати узагальненим багатошаровим персептроном.

# 3. ПРАКТИЧНА ЧАСТИНА

Навчання нейронної мережі буде проводитись на основі завдання про розрахунок середнього балу для студентів, який переведено до системи ECTS згідно таблиці (табл. 1.1).

Таблиця 3.1 *-* Шкала оцінювання в системі ECTS

|  |  |
| --- | --- |
| Середня рейтингова оцінка | Оцінка за шкалою ЕСТS |
| 90 – 100  82 – 89  75 – 81  69 – 74  60 – 68 | А  В  С  D  Е |

Аналіз був проведений на основі балів з 10 предметів: вищої математики, чисельних методів, фізики, української мови, програмування, операційних систем, англійської мови, системного аналізу. Вони представлені середніми балами у вигляді A,B,C,D,E. Навчання проводитиметься на основі 20 спостережень, тобто для визначення підсумкового середнього балу будуть використані оцінки двадцяти студентів.

Розробка даної нейронної мережі буде корисною для визначення підсумкового балу та побудови рейтингового списку для кожного студента за цим параметром.

На рисунку представлено таблицю даних, що являє собою перелік оцінок з десяти предметів для двадцяти студентів та визначений на основі цих даних середній бал згідно системи ECTS (рис. 3.1).

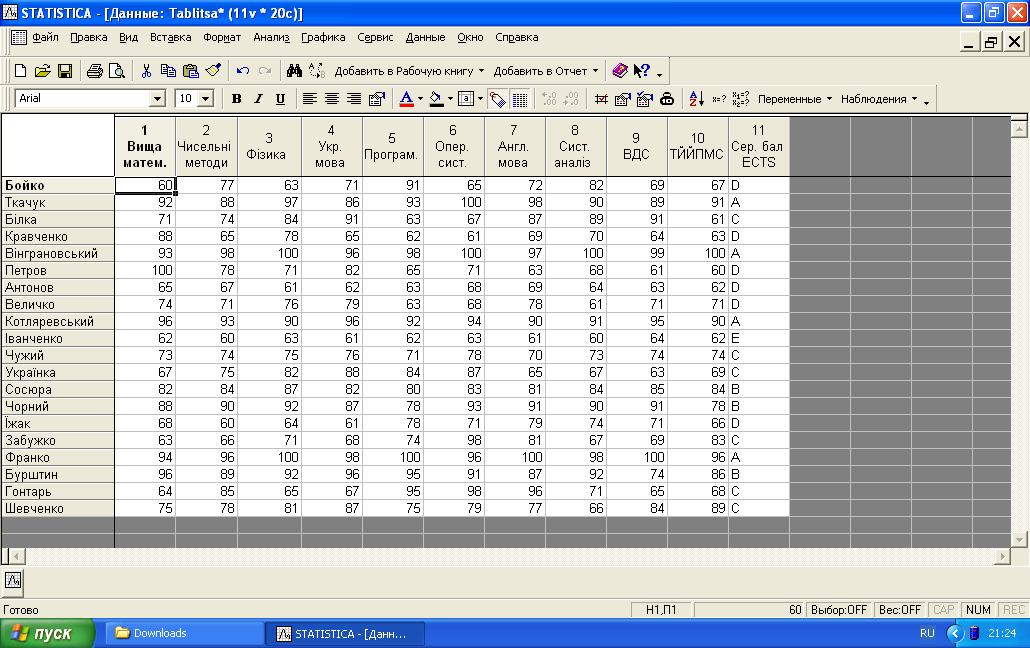


Рисунок 3.1 - Навчальна таблиця

Розглянемо наступні етапи створення нової мережі. Для початку навчання обираємо в меню «Аналіз» пункт «Нейронні мережі».

1. Першим етапом буде навчання за допомогою **Майстра рішень**:

Обираємо інструмент *«*Майстер рішень» з меню і тип задачі «Класифікація», все інше залишаємо без змін, натискаємо ОК (рис. 3.2).

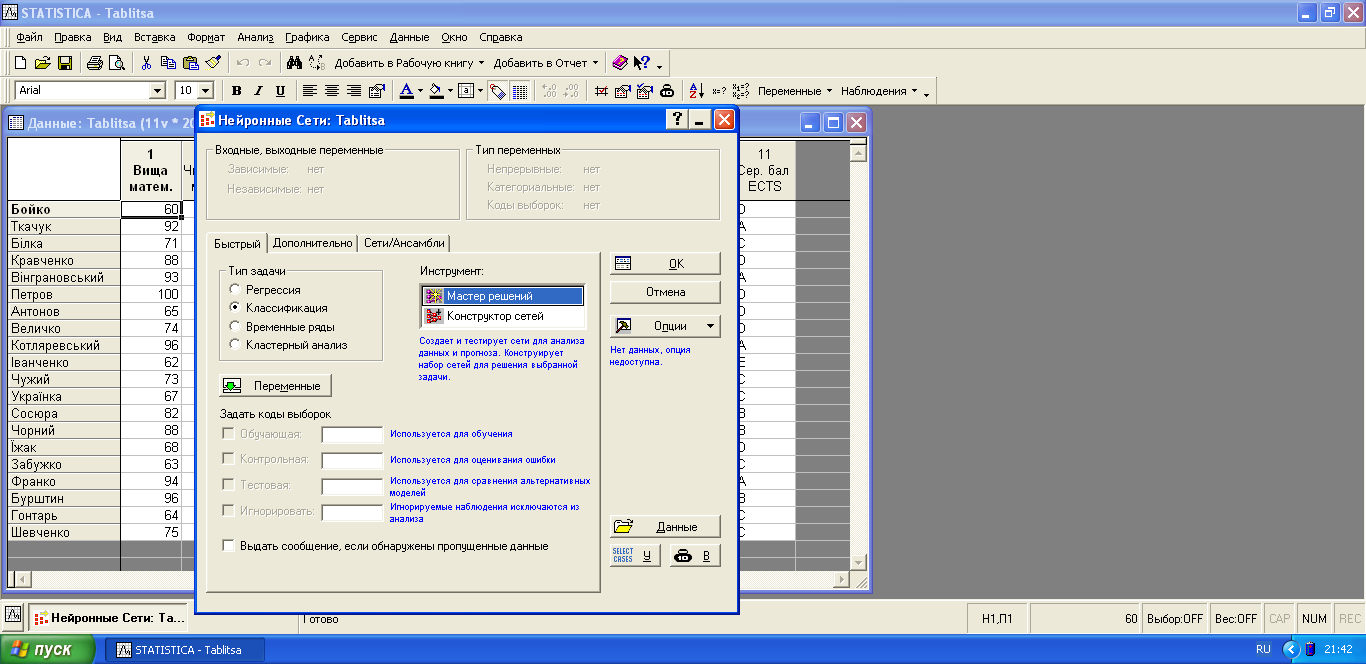


Рисунок 3.2 - Меню вибору типу задачі та інструменту

Необхідно натиснути на кнопку «Змінні» , для того, щоб обрати змінні. Оскільки працюємо з прикладом , що характеризує навчання з вчителем, то залежною змінною позначаю останню колонку, що має назву «Сер.бал. ECTS» – вона є вихідними даними. Решта, 10 змінних, що позначають предмети – характеризуємо вхідними змінними (рис. 3.3).

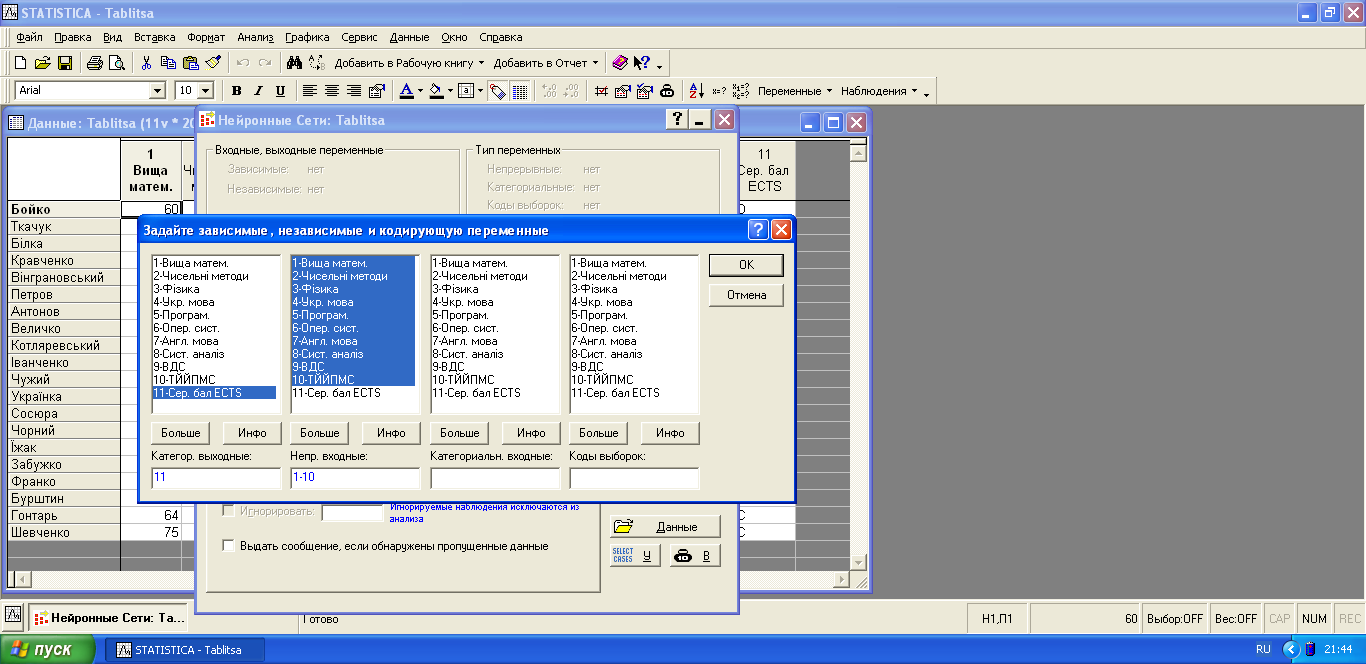


Рисунок 3.3 - Вікно вибору змінних

На вкладці «Швидкий», вікна «Майстер рішень», обираємо кількість мереж, та кількість мереж для збереження. Задамо програмі побудувати і навчити 100 мереж різного типу і вибрати з них 5 найкращих в сенсі критерію, який задається на вкладці «Зберегти» (рис. 3.4).

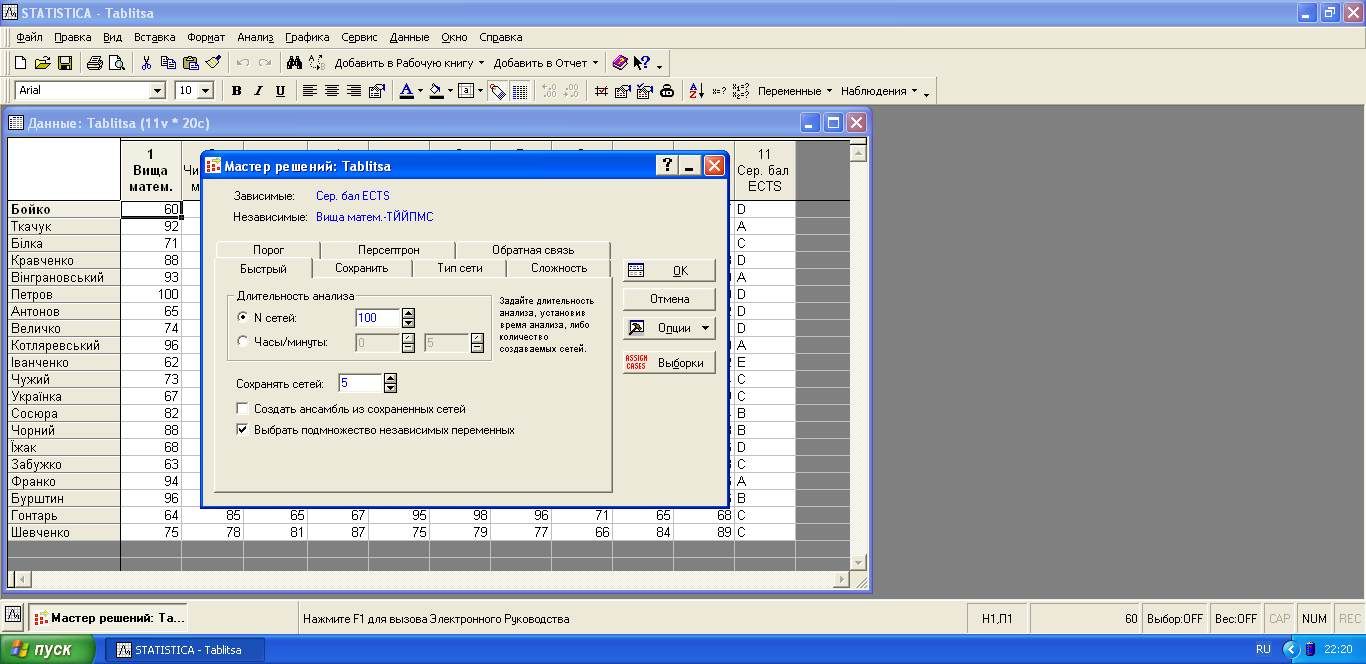
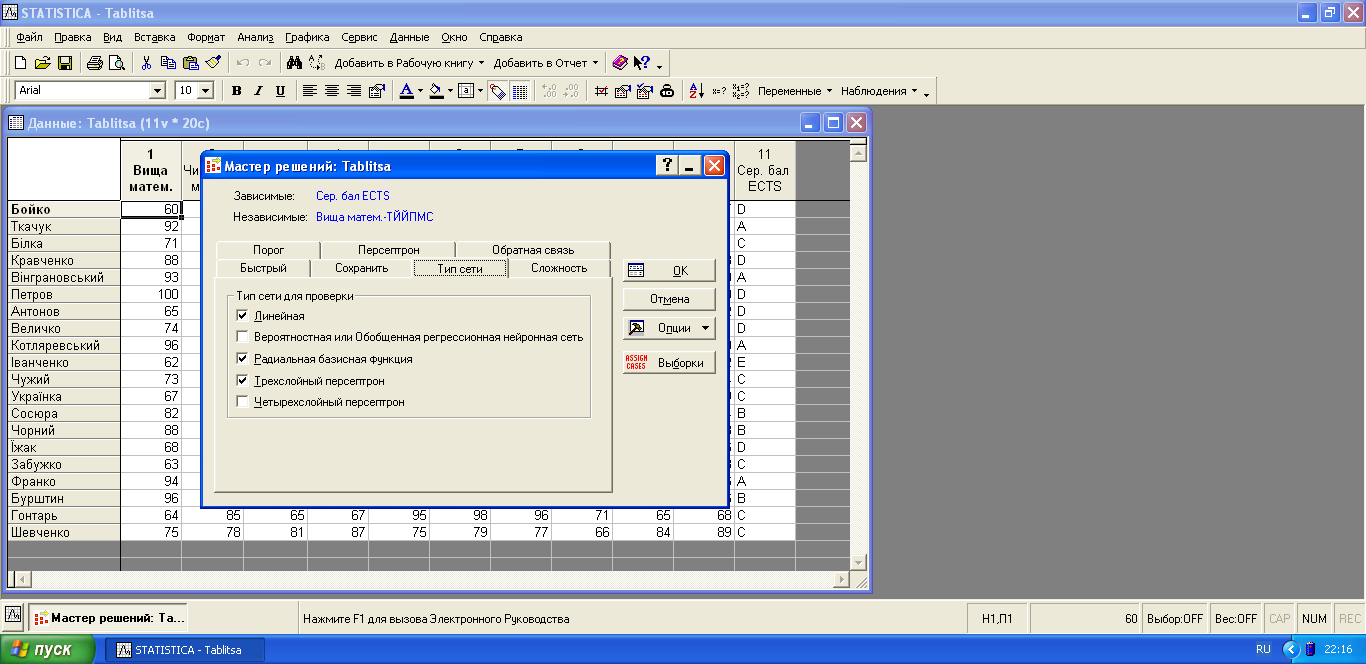


Рисунок 3.4 - Вкладка «Швидкий» діалогового вікна «Майстер рішень»

На вкладці «Тип мережі» обираємо 3 типа мережі для перевірки: «Лінійна», «Радіальна базисна функція» та «Тришаровий персептрон» (рис. 3.5).

 Рисунок 3.5 - Вкладка «Тип мережі» діалогового вікна «Майстер рішень»

На вкладці «Персептрон» в меню «Кодування виходу в класифікації» обираємо «Суму квадратів» (рис. 3.6).

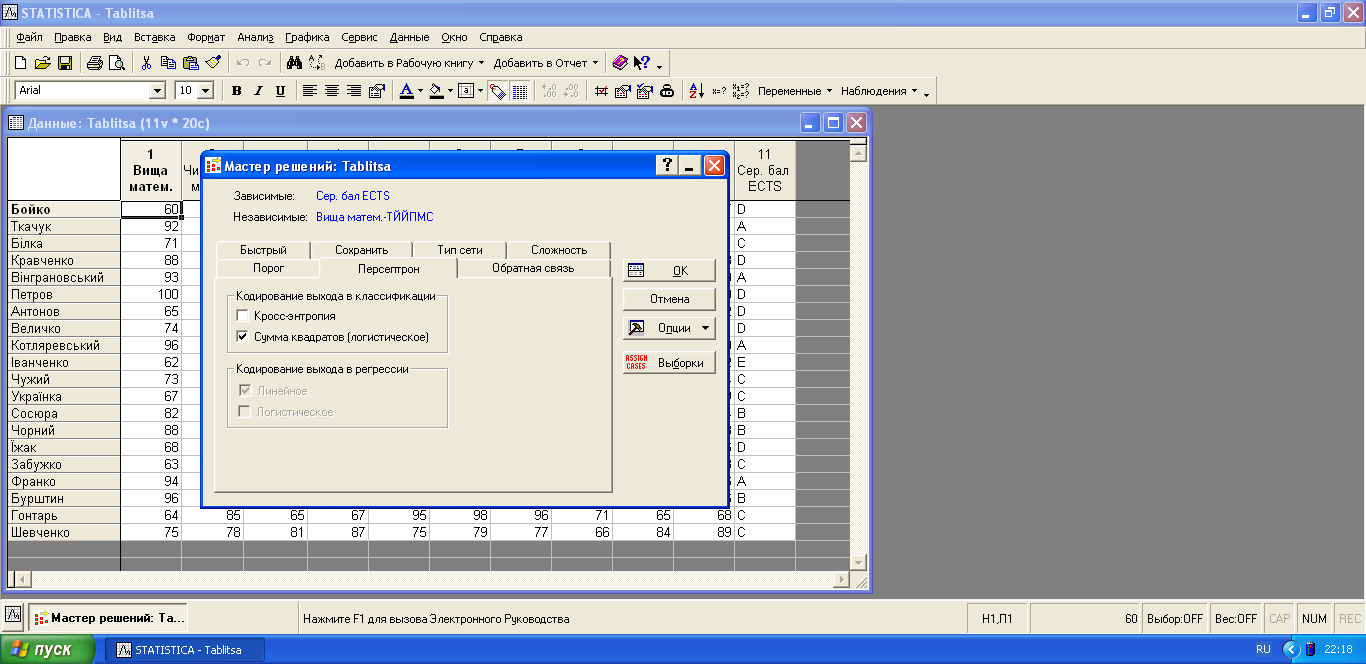


Рисунок 3.6 - Вкладка «Персептрон» діалогового вікна «Майстер рішень»

Дані на всіх інших вкладках залишаємо без змін, натискаємо ОК. Починається процес навчання мережі. Отримуємо такі результати навчання даної нейронної мережі (рис. 3.7).

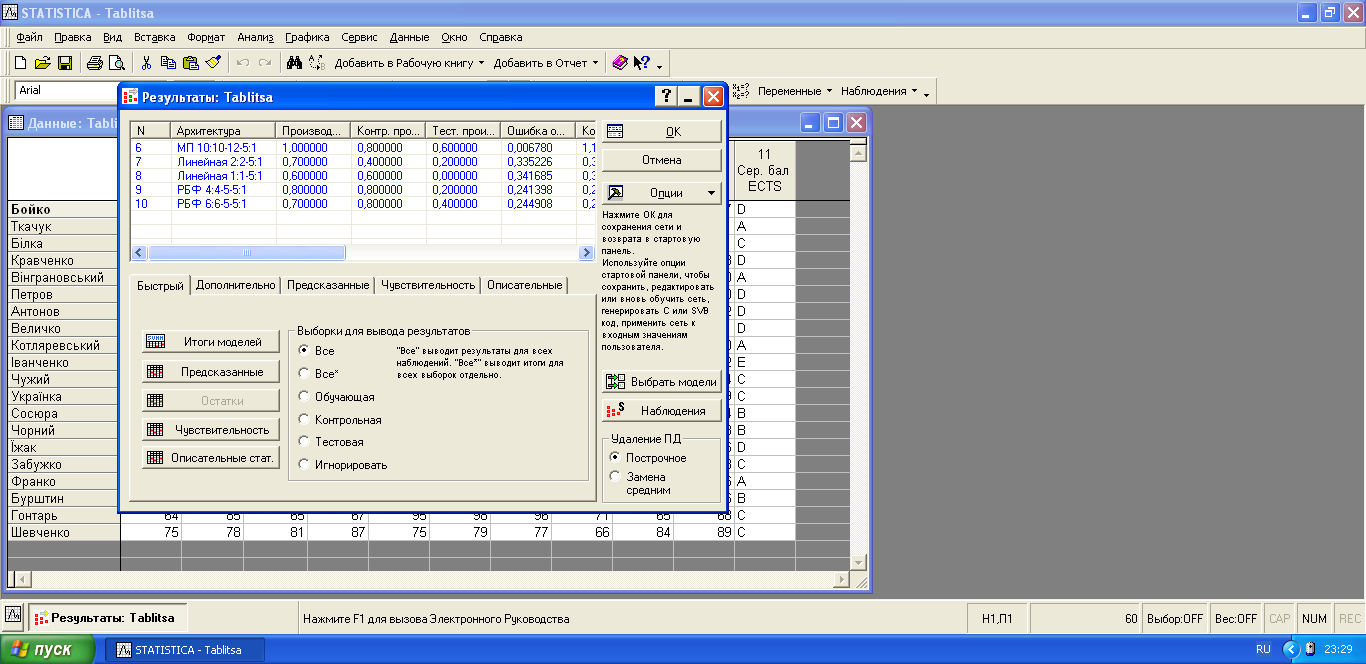


Рисунок 3.7 - Результати навчання нейронної мережі

Обираємо серед запропонованих мереж ту, що навчилась найкраще. В даному випадку це мережа з архітектурою багатошарового персептрону, що має назву МП 10:10-12-5:1. Вона має найвищі показники продуктивності по всім вибіркам (1,0- навчальна, 0,8 – контрольна, 0,6 – тестова), а також найнижчу помилку, що становить 0,00678 (рис. 3.8).

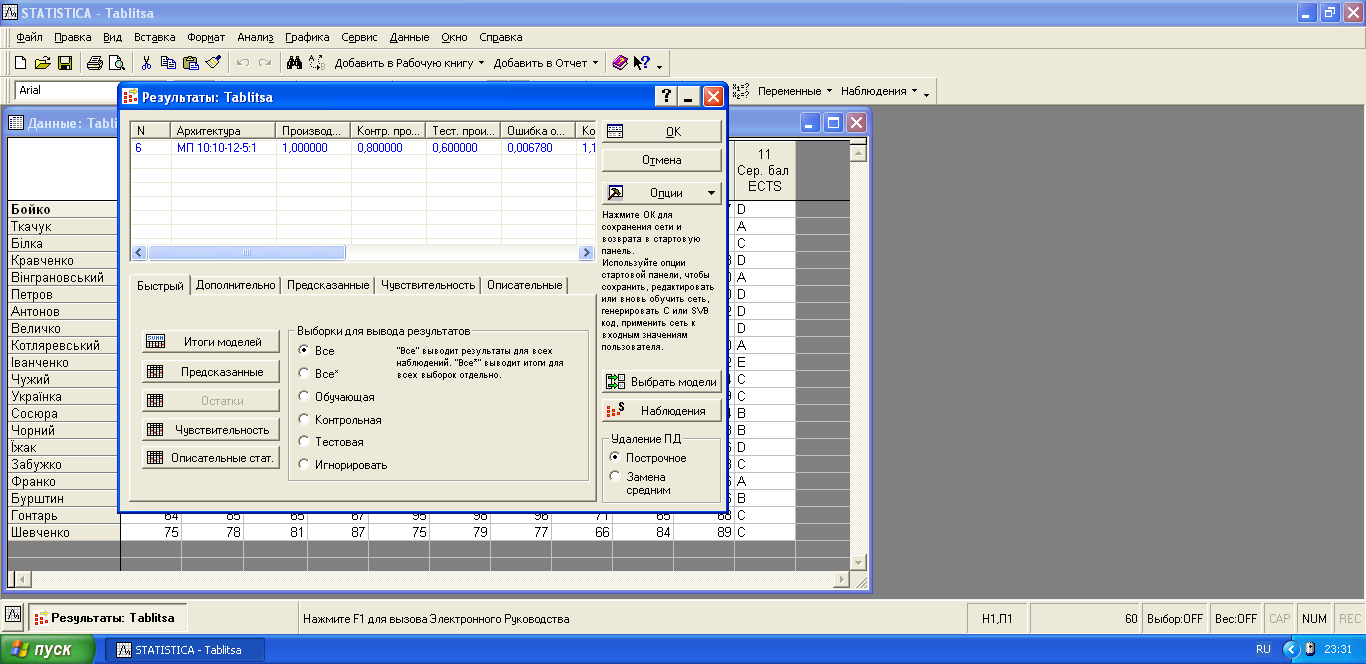


Рисунок 3.8 - Модель мережі, що найкраще навчилася

У вікні «Передбачення» переглядаю результати навчання обраної мережі та порівнюю їх з результатами вихідної змінної. Червоним кольором позначено різницю, помилки обраної мережі з еталонними вихідними даними (рис. 3.9).

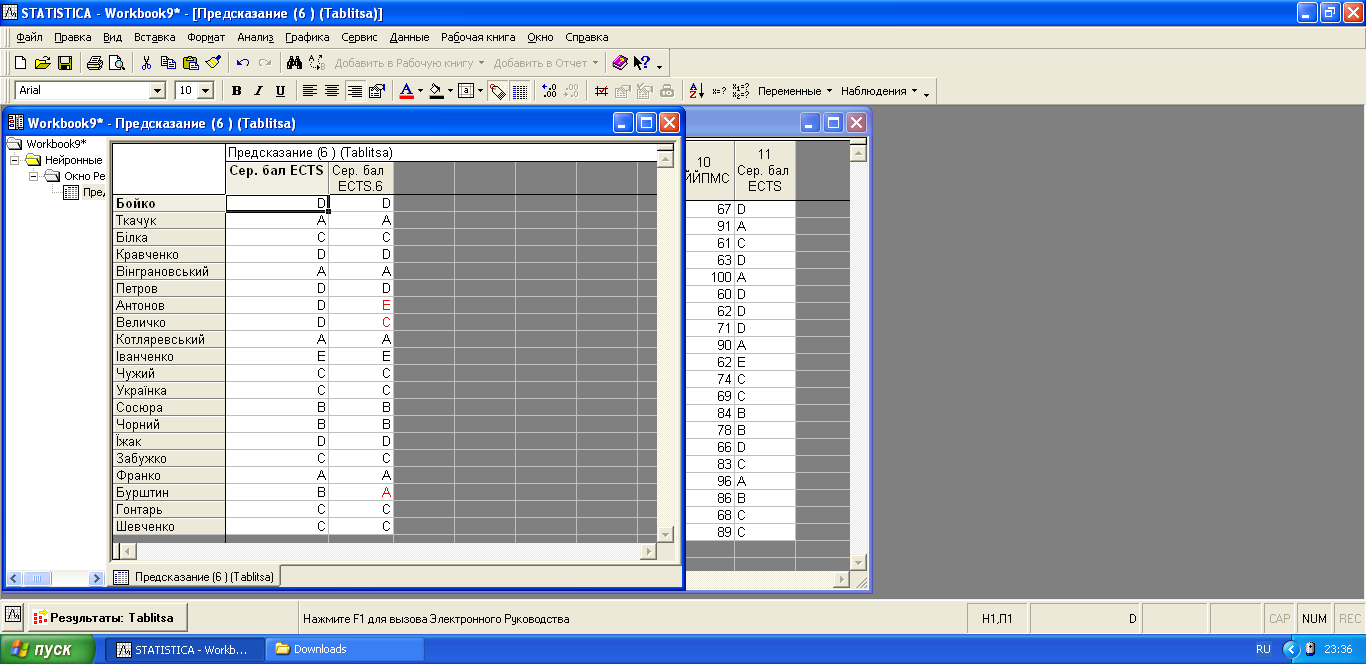


Рисунок 3.9 - Вікно «Передбачення» для обраної моделі нейронної мережі

Для даної задачі вважається, що навчання пройшло успішно, якщо помилка навчання становить менше 20% від загальної кількості спостережень. Враховуючи, що прикладів було досить мало, лише 20, тому результат, що становить 3 помилки, а це 15% , є результатом , що повністю задовольняє умову задачі. Зберігаємо дану нейронну мережу.

Відкриваємо щойно збережену мережу і перевіряємо її працездатність.

На вкладці «Мережі/Ансамблі»обираємо попередньо збережену мережу, яку бажаємо відкрити та запускаємо її у вкладці «Додатково» (рис. 3.10. -3.11).

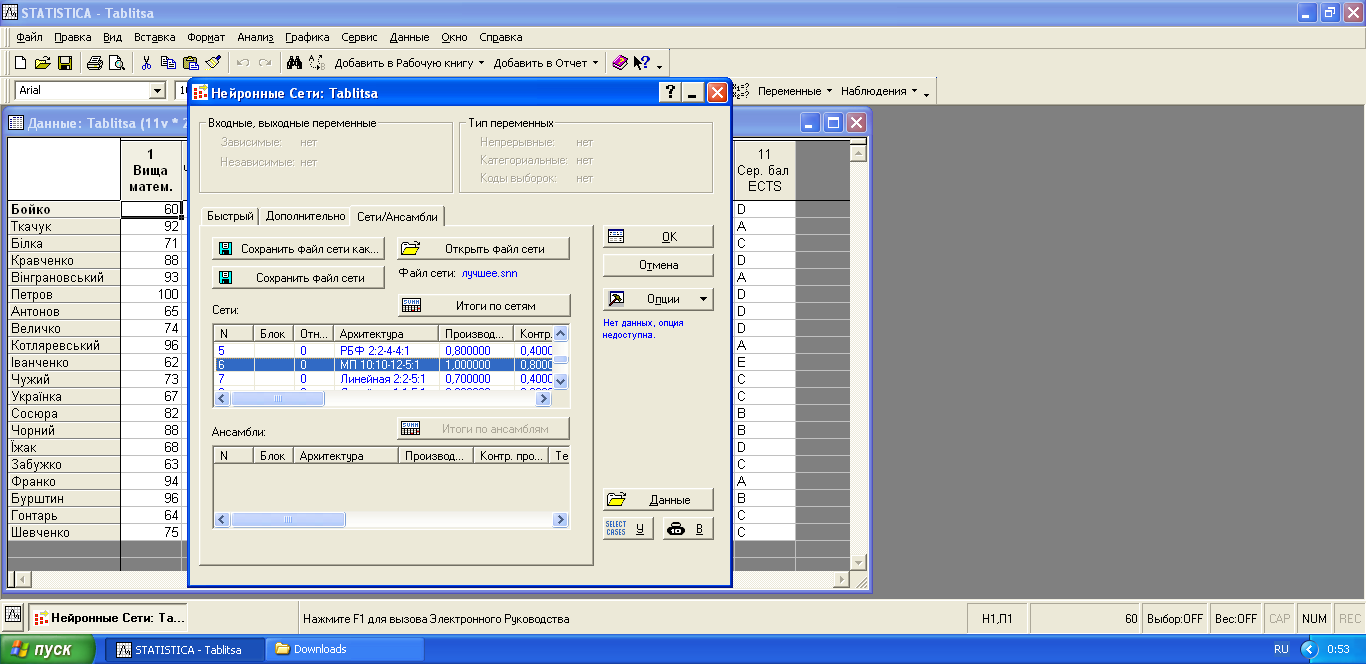


Рисунок 3.10 - Вибір необхідної мережі

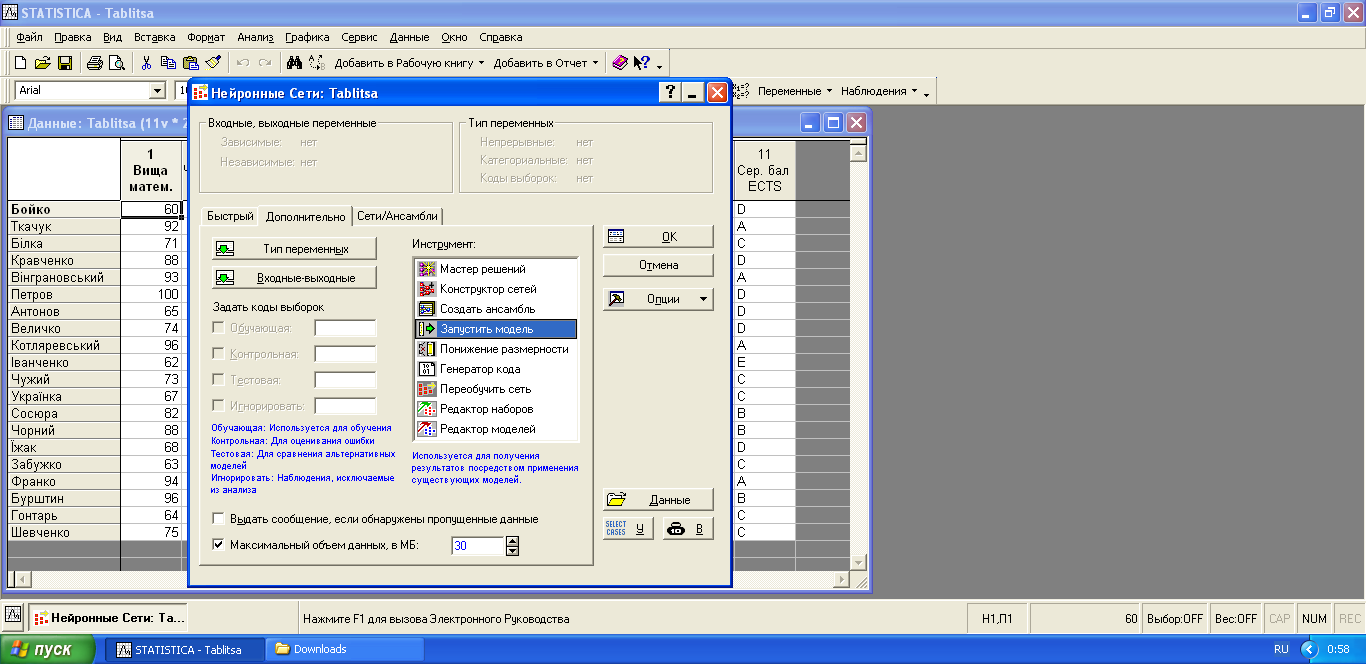


Рисунок 3.11 - Вибір необхідного інструменту для запуску моделі

Після натискання ОК у вікні «Нейронні мережі», відкривається вікно «Прогноз для спостережень користувача» (рис. 3.12). Панель «Значення користувача» дозволяє ввести вхідні дані, в нашому випадку оцінку за кожну дисципліну,аби дізнатися прогнозований результат. (рис. 3.13. – 3.14.)

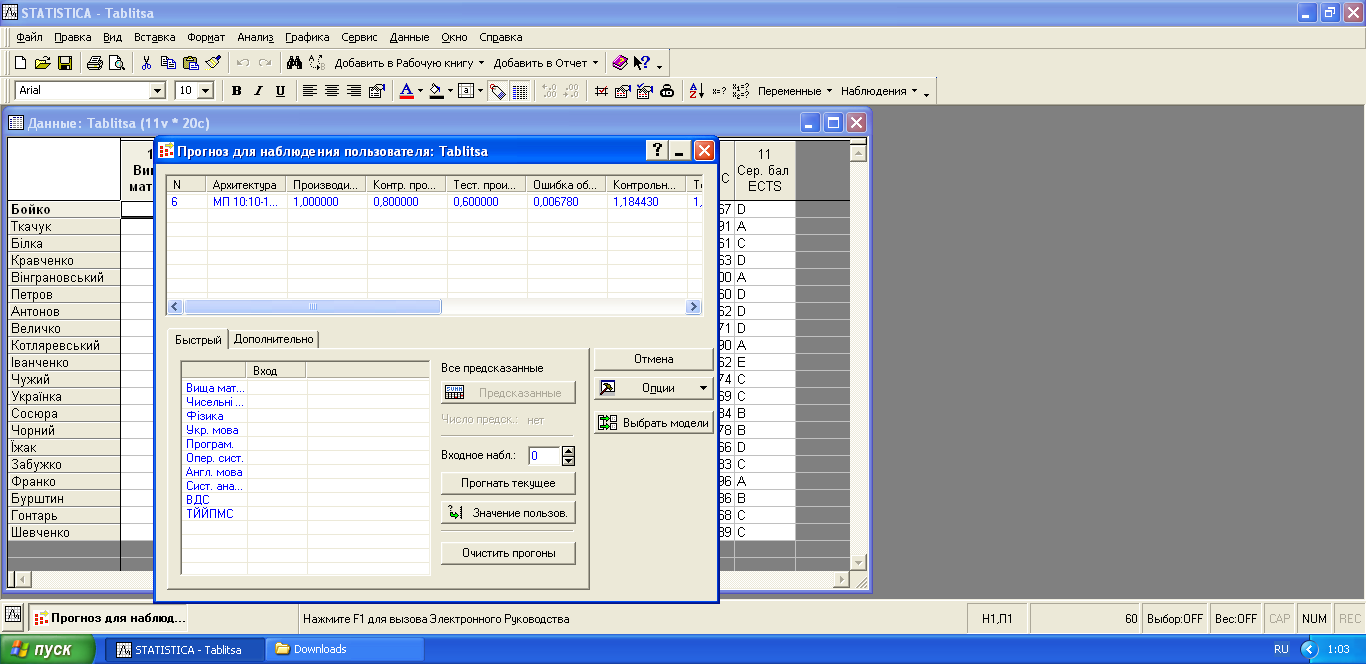


Рисунок 3.12. - Вікно прогнозу для спостережень користувача

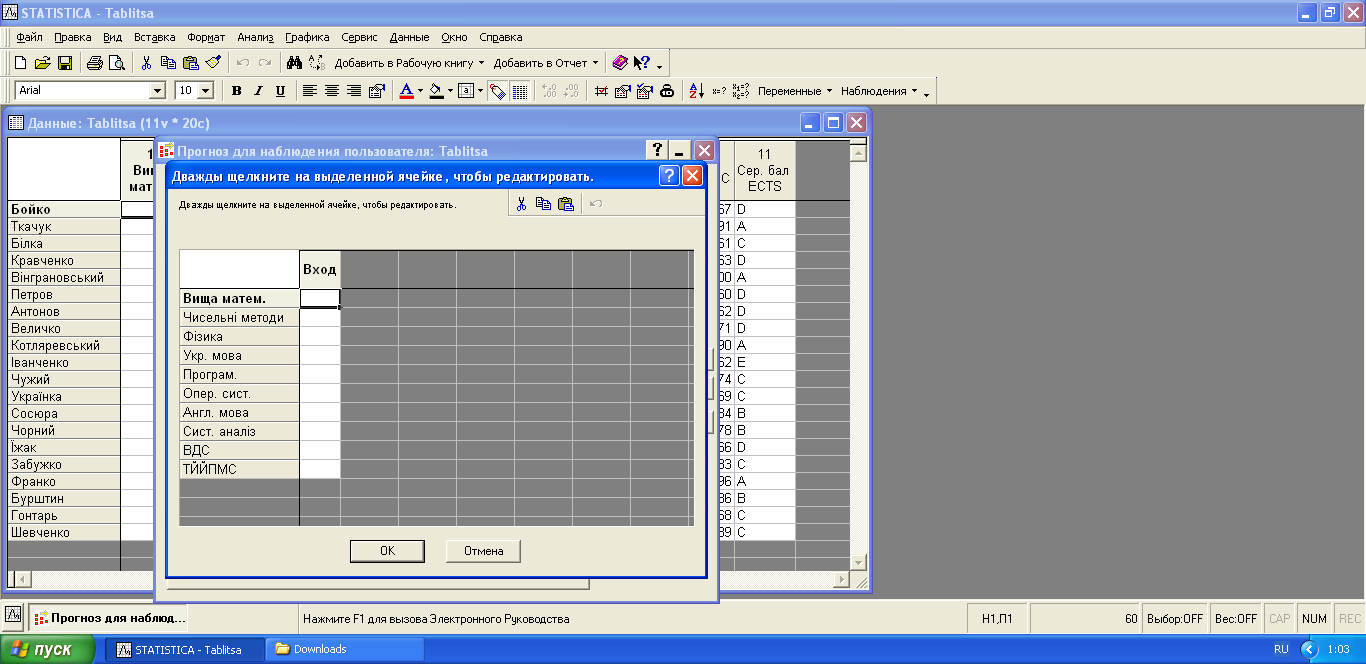


Рисунок 3.13 - Вікно для заповнення нових вхідних даних

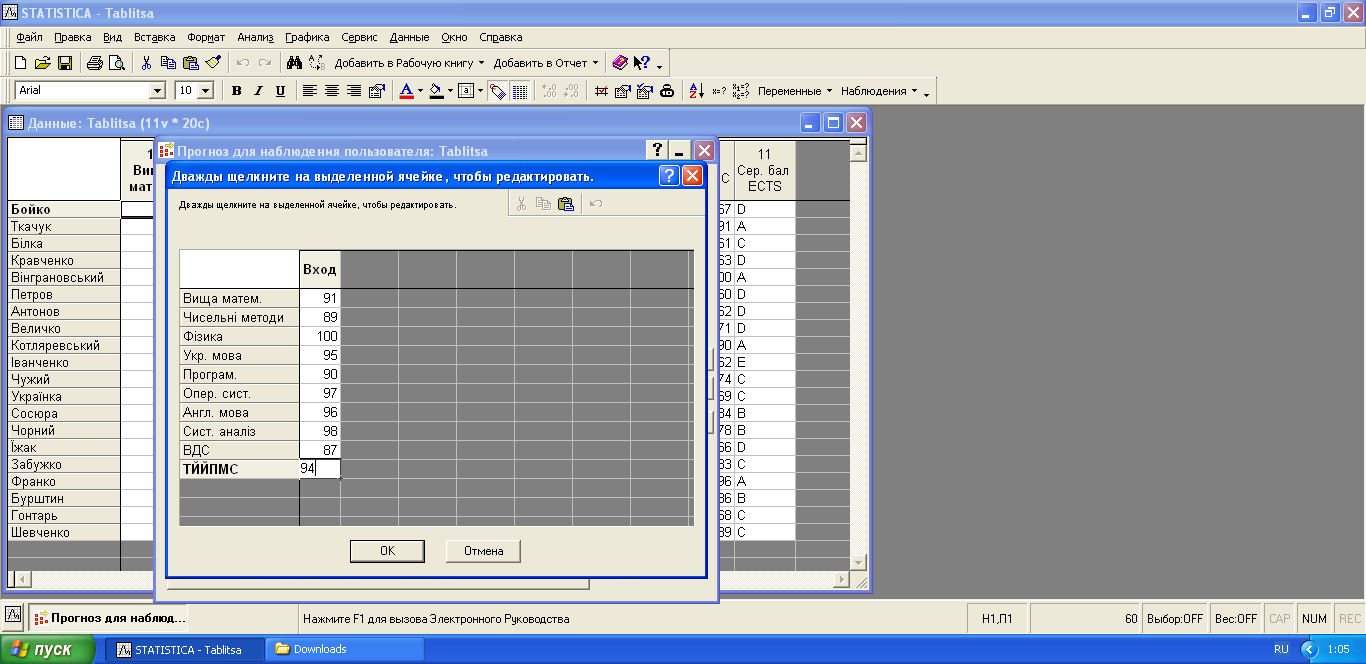


Рисунок 3.14 - Процес заповнення вхідних даних

Після заповнення вікна для вхідних даних отримуємо результат, неозброєним оком видно, що прогнозованим середнім балом за шкалою ECTS повинна бути оцінка «А» (рис. 3.15).

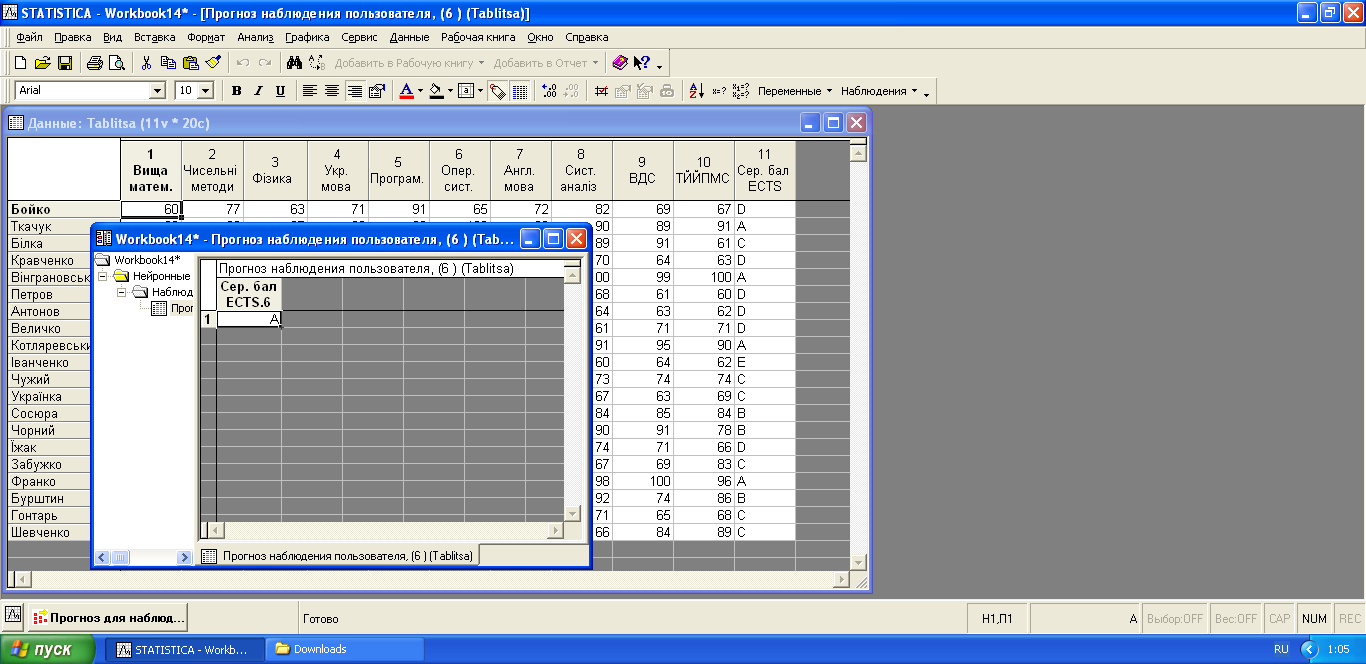


Рисунок 3.15 - Отриманий прогноз середнього балу

Результатом прогнозування спостережень користувача нейронною мережею є саме оцінка «А», тому дана модель є працездатною і її можна використовувати для вирішення даної задачі.

1. Виконаємо навчання мережі за допомогою **Конструктора мережі**.

Цей пункт дозволяє змінювати параметри навчання мережі, таким чином обираючи найкращий варіант .

Для виконання завдання обираємо інструмент «Конструктор мережі», а також тип задачі «Класифікацію» (рис. 3.16).

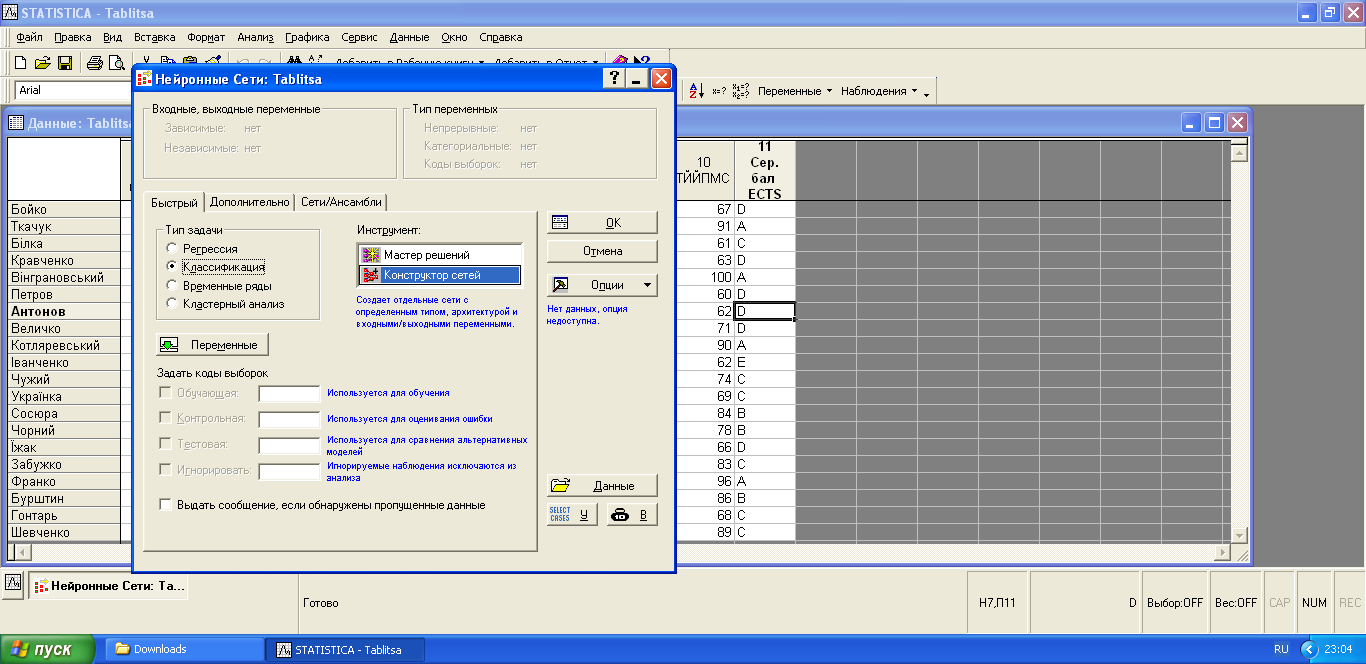


Рисунок 3.16 - Конструктор мереж

Обираємо змінні аналогічним чином (рис. 3.17).

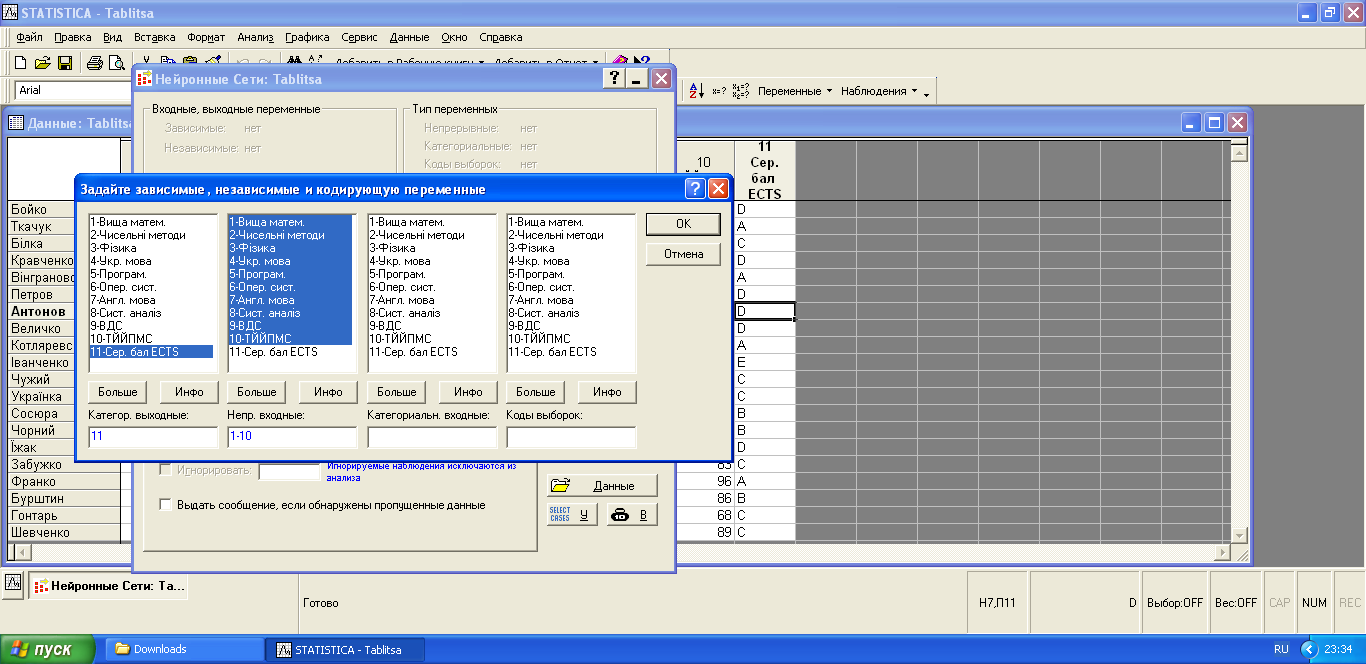


Рисунок 3.17 - Вікно вибору змінних

На вкладці «Швидкий» обираємо архітектуру нейромережі «Багатошаровий персептрон» (рис. 3.18).

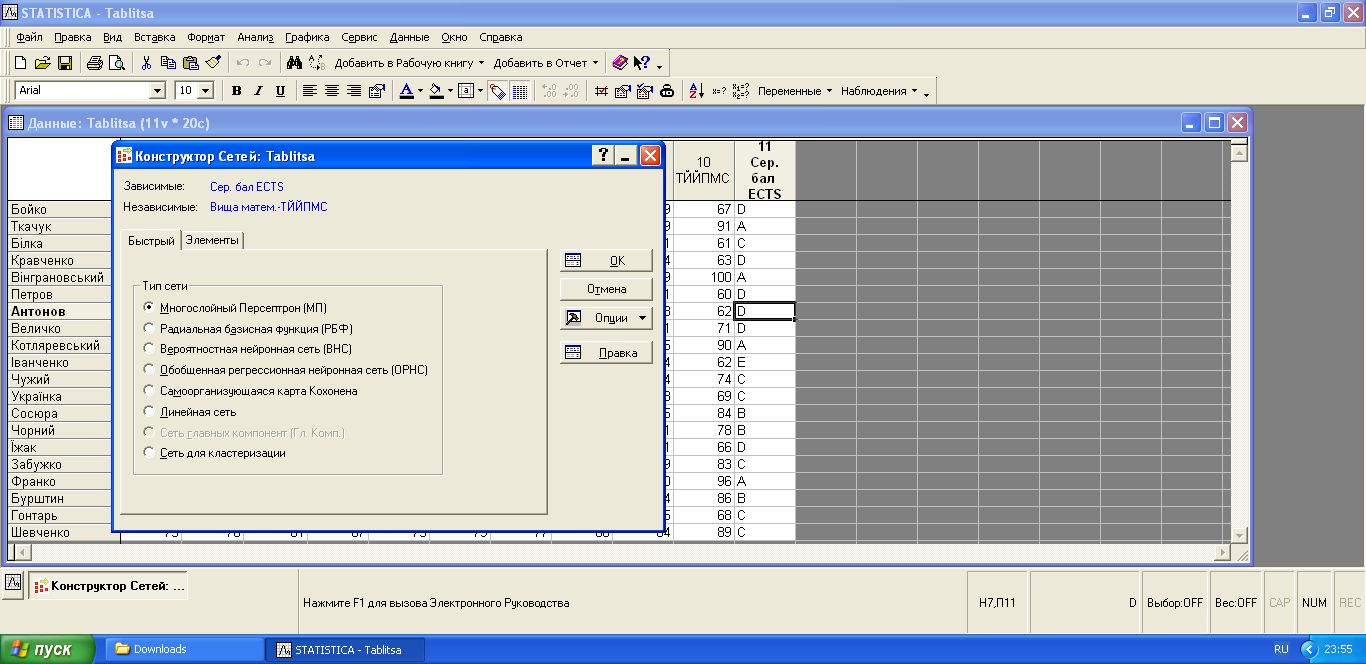


Рисунок 3.18 - Вкладка «Швидкий» діалогового вікна «Конструктор»

На вкладці «Елементи» обираємо кількість прихованих шарів 1, Кількість елементів в прихованому шарі – 10, функцію класифікації – «Крос-ентропію»(рис. 3.19).

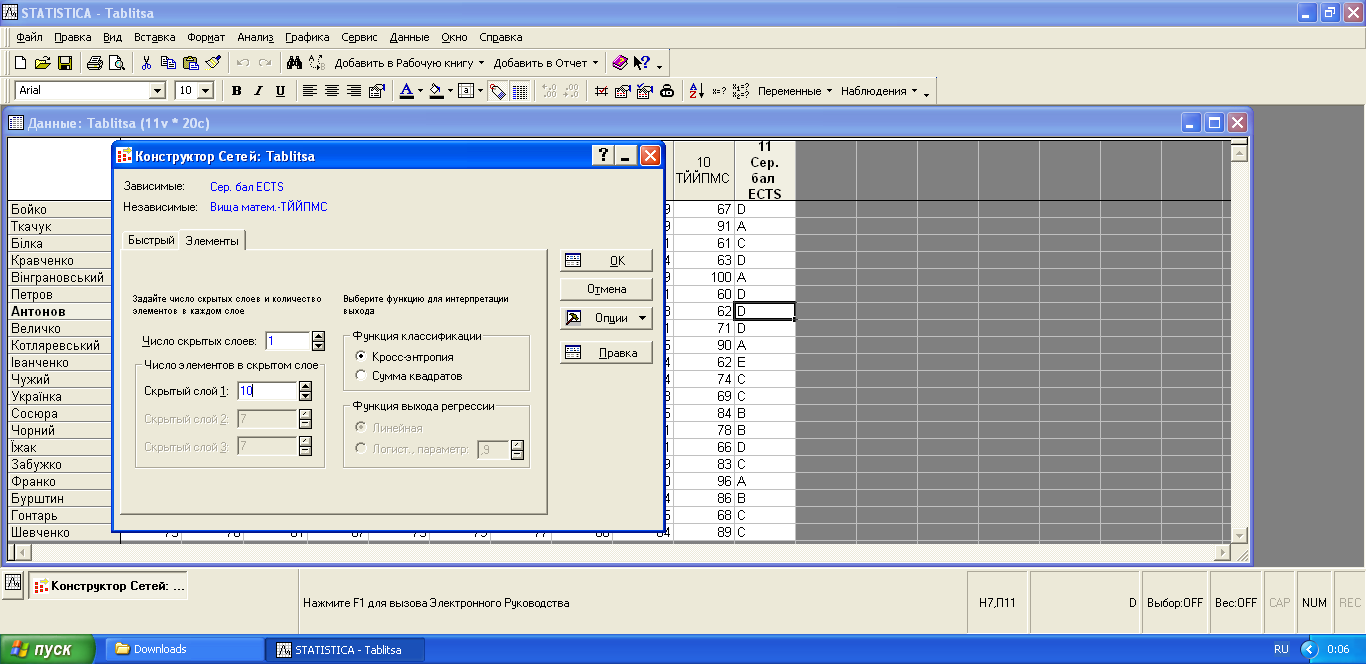


Рисунок 3.19 - Вкладка «Елементи» вікна «Конструктор мереж»

На вкладці «Швидкий» вікна «Багатошаровий персептрон» обираємо тільки перший етап. Висока швидкість навчання дозволяє швидко виконати весь алгоритм, але в деяких випадках можуть бути отримані нестабільні результати. Кількість епох визначаємо за принципом: чим менша швидкість навчання, тим більша кількість епох. Обираємо кількість епох 600 (рис. 3.20).

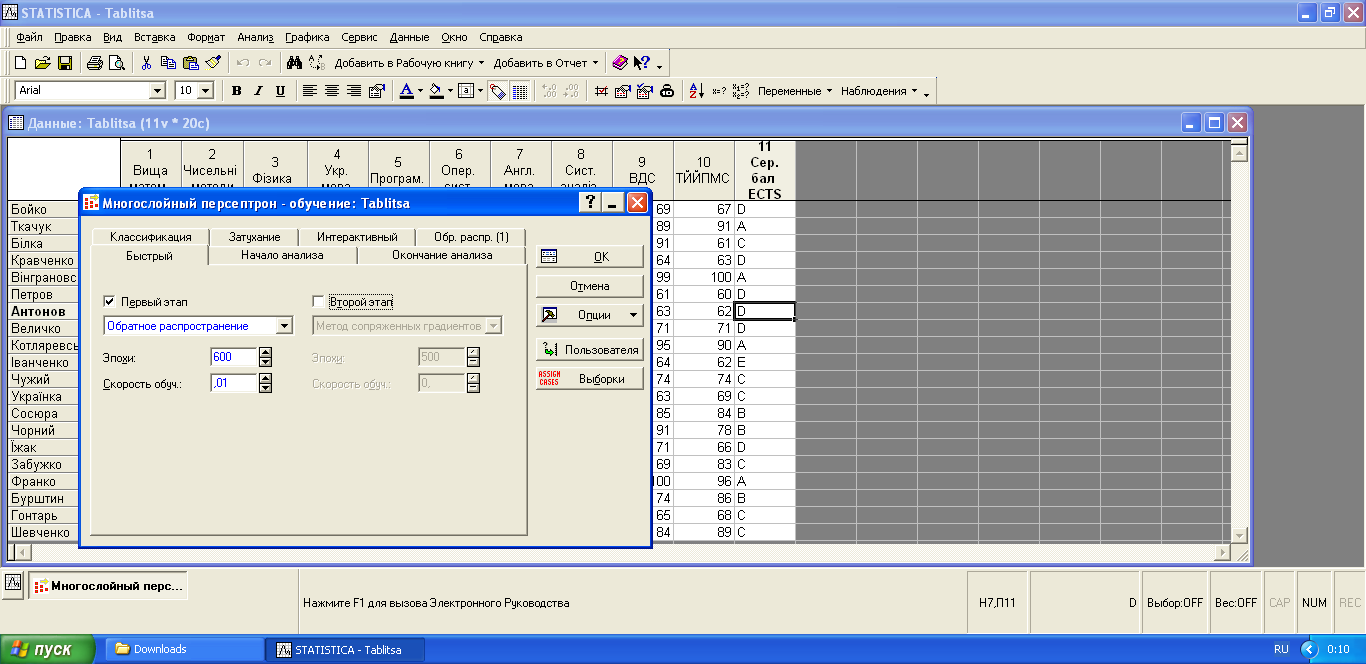


Рисунок 3.20 - Вкладка «Швидкий» діалогового вікна «Багатошаровий персептрон»

У вкладці «Класифікація» встановлюємо довірчі пороги. Оскільки раніше зазначалося похибка навчання не має перевищувати 20%, визначаємо пороги прийняття і відхилення відповідно як 0,8 та 0,2. Це означає, що після навчання нейромережевий пакет має представити нам модель мережі, яка має продуктивність не менше 0,8 (рис. 3.21).

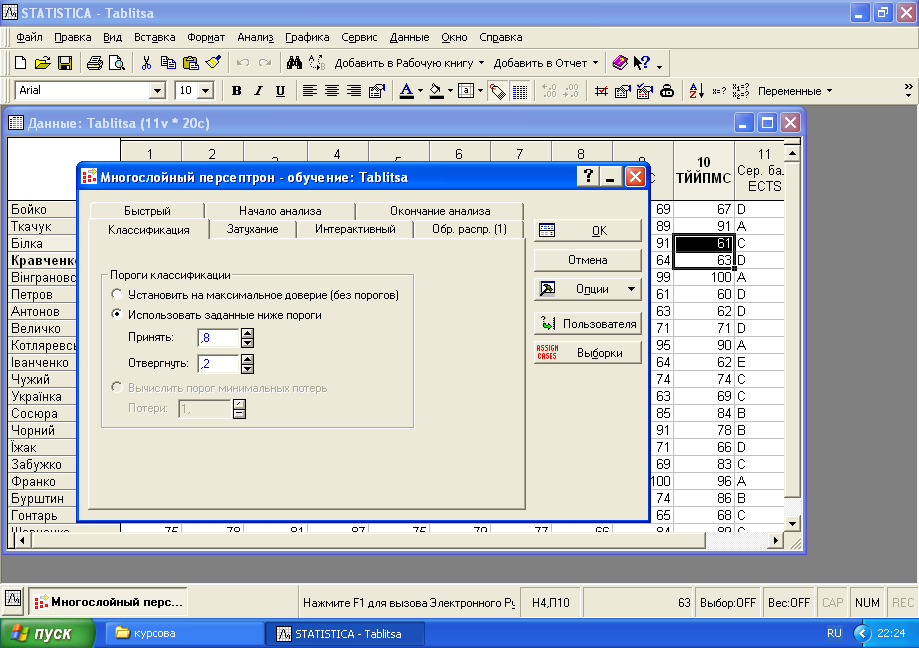


Рисунок 3.21 - Вкладка «Класифікація» діалогового вікна «Багатошаровий персептрон»

Все інше залишаємо без змін, натискаємо ОК. Починається навчання. Через декілька секунд з’являється результат (рис. 3.22).

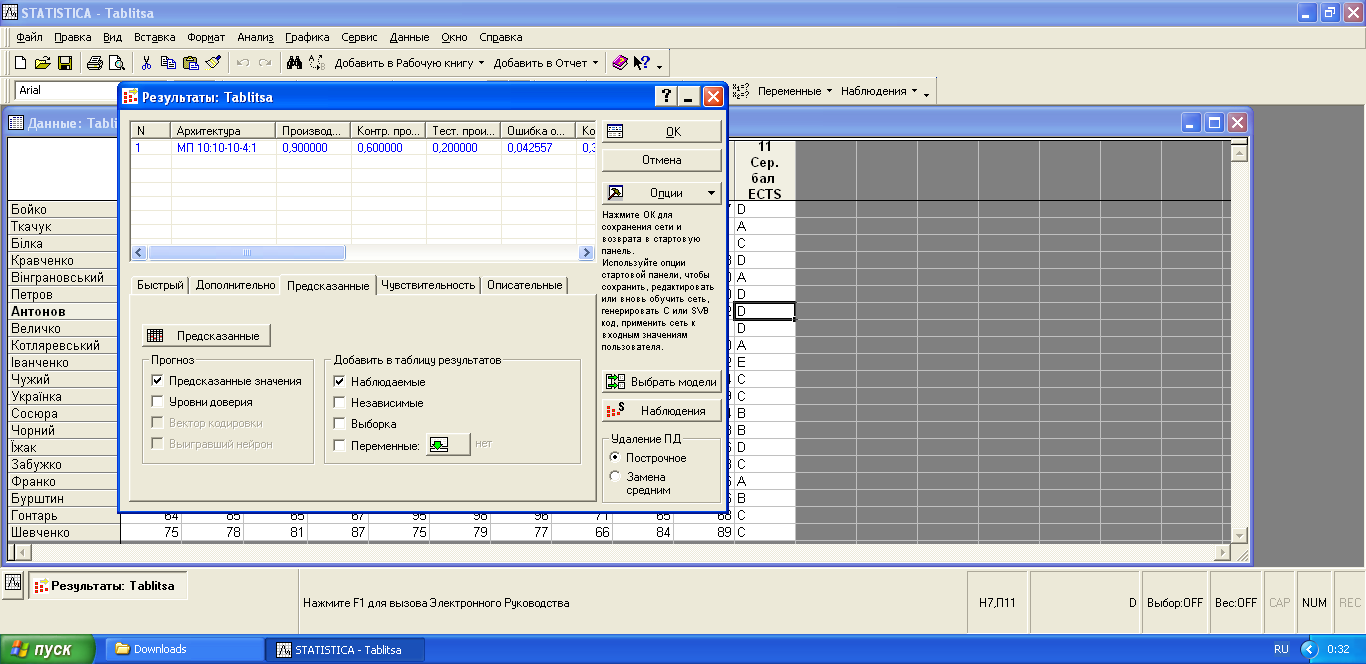


Рисунок 3.22 - Результати навчання нейронної мережі

Після навчання, статистичний пакет сам визначає мережу, що навчилась найкраще. В даному випадку, це мережа з архітектурою багатошарового персептрону, що має назву МП 10:10-10-4:1. Вона має такі показники продуктивності по всім вибіркам (0,9 – навчальна, 0,6 – контрольна, 0,2 – тестова) , а також помилку, що становить 0,042557 .

У вікні «Передбачення» переглядаю результати навчання обраної мережі та порівнюю їх з результатами вихідної змінної. Червоним кольором позначено різницю, помилки обраної мережі з еталонними вихідними даними (рис. 3.23).

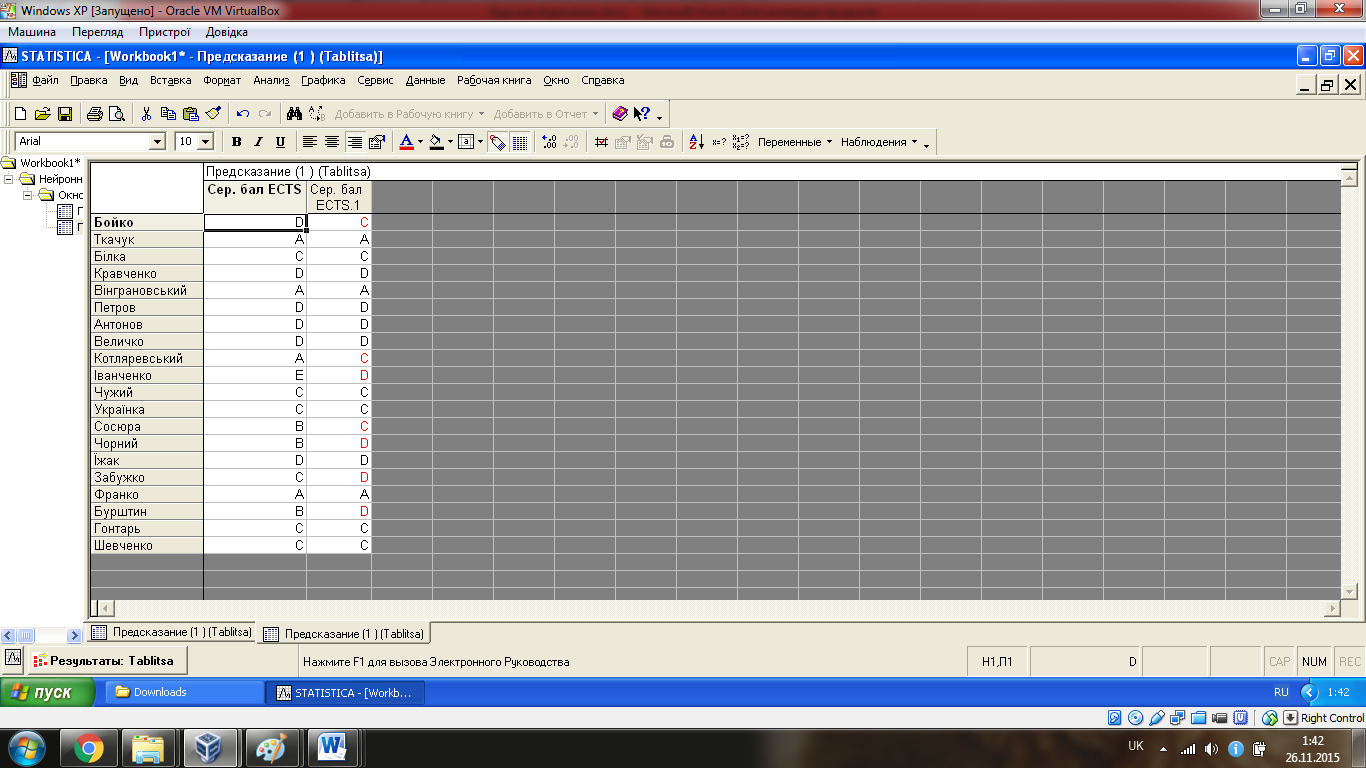


Рисунок 3.23 - Вікно «Передбачення» для даної моделі нейронної мережі

Для даної задачі вважається, що навчання пройшло успішно, якщо похибка навчання становить менше 20% від загальної кількості спостережень. Враховуючи, що прикладів було досить мало, лише 20, тому результат, що становить 7 помилок, а це 35% , є результатом , що перевищує встановлену норму і є незадовільним для даної задачі. Зберігаємо дану нейронну мережу.

Відкриваємо щойно збережену мережу і перевіряємо її працездатність(рис. 3.24. – 3.25).

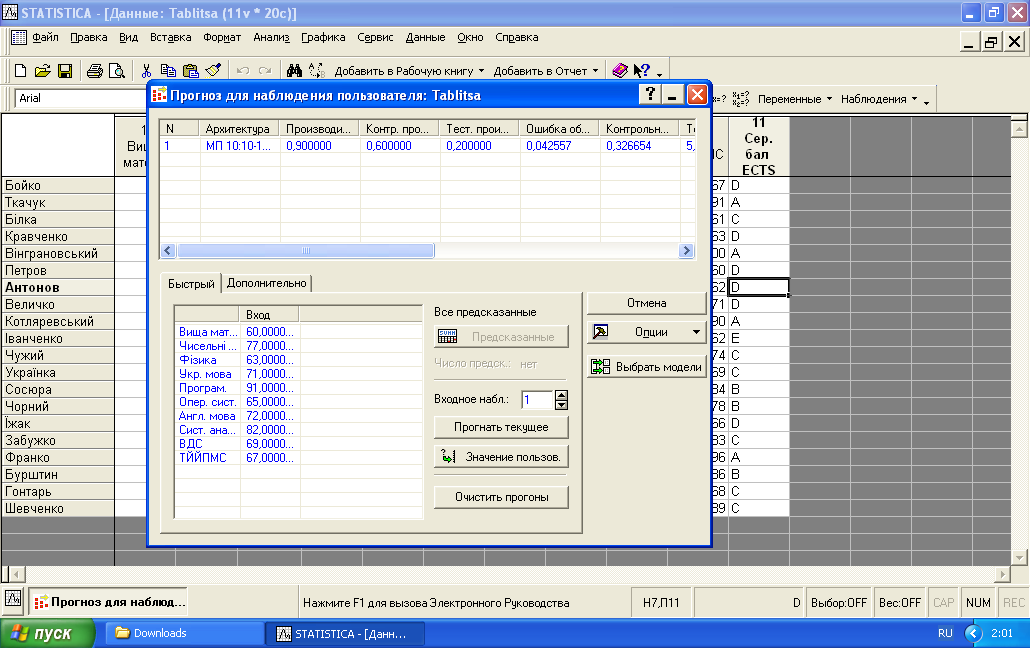


Рисунок 3.24 - Вікно прогнозу для спостережень користувача

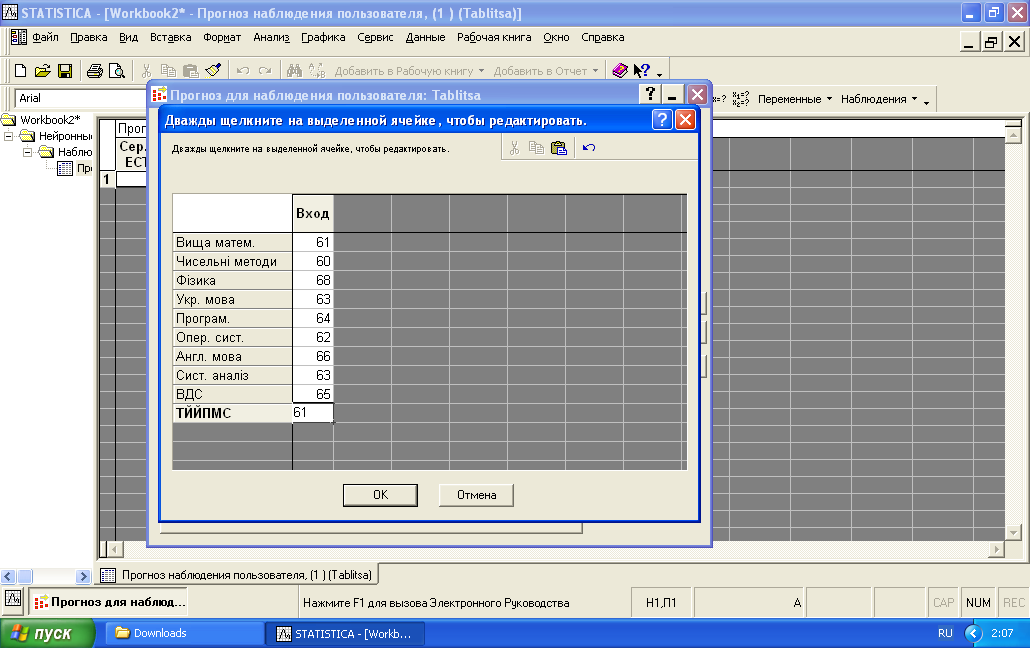


Рисунок 3.25 - Процес заповнення вхідних даних

Після заповнення вікна для вхідних даних отримуємо результат, неозброєним оком видно, що прогнозованим середнім балом за шкалою ECTS повинна бути оцінка «Е» (рис. 3.26).

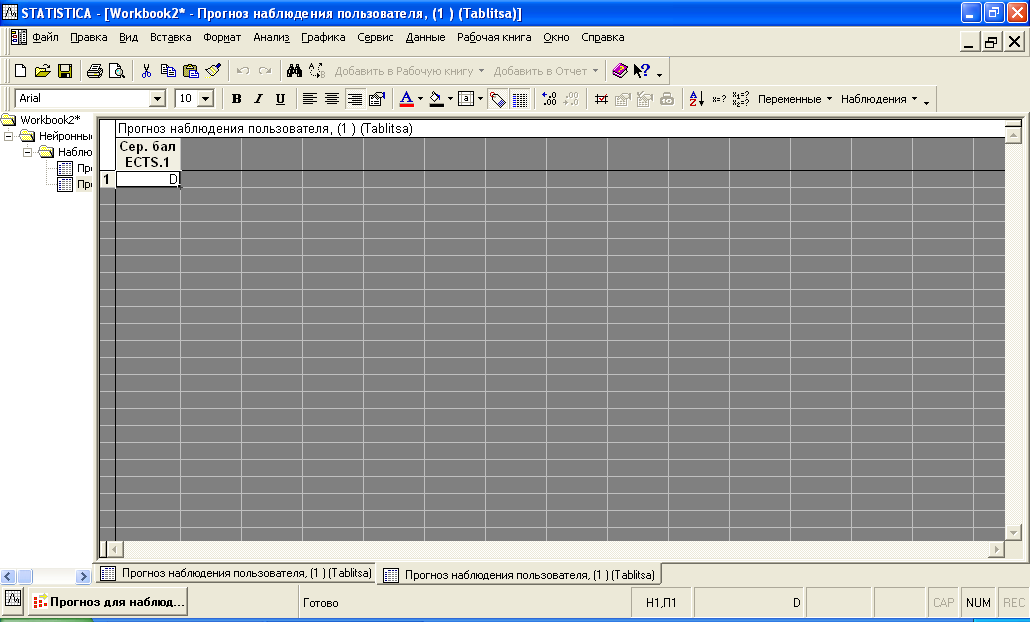


Рисунок 3.26 - Отриманий прогноз середнього балу

Результатом прогнозування спостережень користувача нейронною мережею є оцінка «D», що не відповідає істинності, тому дану модель не можна вважати працездатною. Мережа недостовірно визначає середній бал, помиляючись на одну позицію, це пояснюється завеликою кількістю похибок.

**Висновок**

Під час виконання роботи, за допомогою нейромережевого програмного продукту StatSoft STATISTICA 6.1, було визначено в чому полягає навчання як процес реалізації штучного інтелекту.

Було проведено декілька навчань, використовуючи інструменти «Майстер мереж» та «Конструктор мереж»,а також різні конфігурації. Після навчань мереж для задачі визначення середнього балу за шкалою ЕСTS було отримано такі результати.

Для майстру мереж – нейромережею, що має найвищі показники продуктивності по всім вибіркам (1,0- навчальна, 0,8 – контрольна, 0,6 – тестова) , а також найнижчу помилку, що становить 0,00678 стала модель з архітектурою багатошарового персептрону, що має назву МП 10:10-12-5:1. Передбачені результати встановили, що похибка навчання для неї становить 15% , що не перевищує критичного рівню в 20%, тому мережа є працездатною.

Для інструменту конструктора нейромережею, що має найвищі показники продуктивності по всім вибіркам (0,9 – навчальна, 0,6 – контрольна, 0,2 – тестова), а також найнижчу помилку, що становить 0,042557 стала модель з архітектурою багатошарового персептрону, що має назву МП 10:10-10-4:1. Передбачені результати встановили, що похибка навчання для неї становить 35% , що перевищує дозволені 20%, тому мережа не є працездатною.

Це дає підстави говорити, що для вирішення задачі класифікації для визначення підсумкового середнього балу студентів доцільніше використовувати інструмент « Майстер мереж», а ніж «Конструктор мереж».

# Список використаних джерел

1. Нейронные сети. STATISTICA Neural Networks: Методология и технология современного анализа даннях/ Под редакцией В.П. Боровикова. – 2-е изд., перераб. и доп. – М.: Горячая линия – Телеком, 2008. – 392 с., ил.
2. StatSoft Inc.: Електронний підручник зі статистики. [Електронний ресурс] – Режим доступу до інформації: [http://www.statsoft.ru/home/textbook / default.htm.](http://www.statsoft.ru/home/textbook%20/%20default.htm.)
3. Відеоролики по нейромережам. [Електронний ресурс] – Режим доступу до інформації: <http://www.statsoft.ru/tv/screen-video/neural-networks/>
4. Обучение нейронной сети. [Електронний ресурс] – Режим доступу до інформації:<http://it-karkas.com.ua/uploads/files/public-folder/AI/Lec_SAI_14.pdf>
5. Штучна нейронна мережа. [Електронний ресурс] – Режим доступу до інформації:<https://uk.m.wikipedia.org/wiki/%D0%A8%D1%82%D1%83%D1%87%D0%BD%D0%B0_%D0%BD%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D0%B0_%D0%BC%D0%B5%D1%80%D0%B5%D0%B6%D0%B0>
6. Руденко О. Г., Бодянський Е. В. Штучні нейронні мережі: навч. посіб. – Харків: ТОВ “Компанія СМІТ”, 2005. – 408 с.
7. Медведев В. С., Потемкин В. Г. Нейронные сети. МАТЛАВ 6 / под общ. ред. В. Г. Потемкина. – М.: Диалог. – МИФИ, 2002. – 489 с.
8. Акулов П.В. Решение задач прогнозирования с помощью нейронных сетей / Акулов Павел Владимирович [Електронний ресурс]  – Режим доступу: www. .dgtu.donetsk.ua
9. Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, An introduction to statistical learning. – Springer Science+Business Media New York, 2013. – 426 p.
10. Круглов В. В., Борисов В. В. Искусственные нейрон- ные сети. Теория и практика. – 2-е изд. – М.: Горячая линия-Телеком, 2002. – 382 с.
11. Уоссермен Ф. Нейро- компьютерная техника: Теория и практика / пер. с англ. Ю. А. Зуев, В. А. Точенов. – 1992. – 184 с.
12. ВИКОРИСТАННЯ НЕЙРОННИХ МЕРЕЖ – ПЕРСПЕКТИВНА СФЕРА НАУКИ І СУСПІЛЬСТВА Автор: Дмитро Коцур. [Електронний ресурс]  – Режим доступу: <http://oldconf.neasmo.org.ua/node/139>
13. Оссовский С. Нейронные сети для обработки информации / Станислав Осовский. Пер. с польского И.Д. Рудинского. — М.: Финансы и статистика, 2002. - 344 с.
14. Кальченко Д. Нейронные сети: на пороге будущого / Даниил Кальченко // КомпьютерПресс -  2005. - N1. [Електронний ресурс]  – Режим доступу: htt //www.compr.ru
15. Круг П.Г. Нейронные сети и нейрокомпьютеры: Учебное пособие по курсу «Микропроцессоры» / П.Г. Круг – М.: Издательство МЭИ, 2002. – 176 с.

**(ДОДАТОК)**

**КОД ПРОГРАМИ ІНТЕЛЕКТУАЛЬНОГО КЛАСИФІКАТОРА**

\* standard includes. math.h needed for exp() function. \*/

#include <stdio.h>

#include <math.h>

#include <string.h>

#include <stdlib.h>

#ifndef FALSE

#define FALSE 0

#define TRUE 1

#endif

#define MENUCODE -999

static double listing30Thresholds[] =

{

/\* layer 1 \*/

5.839641626784271, -2.1404928890345754, -1.400930994638119, 0.50760216033781003,

-1.5410350613098904, 0.81427187961996184, -0.7167720180131173, -0.53967107197998987,

0.36259948030792533, 1.8673200496956961,

/\* layer 2 \*/

-4.3541719847929539, 2.2284277045166028, 0.00070368704958627057, 1.201700749476746

};

static double listing30Weights[] =

{

/\* layer 1 \*/

11.517039038679329, -1.5238684323023646, -1.5292308841266609, -1.6815582243522758,

2.8555064251302444, -0.73675130382663789, -1.0838486171759867, -2.7752258617840364,

-2.1047510790467725, -1.8976618847439766, -1.8927340790458658,

-13.044768804998759, 0.11830256953879331, -0.083894454447756214, 0.013483885417538361,

-3.6241072732565391, -0.576153169216571, 0.18638151841551626, 0.78396103751292123,

0.69959705798629312, 0.61056415574201406, 0.70849664001685697,

0.060665690504092255, 0.84225948531133921, -0.39685602862575486, -0.08485173612725759,

-0.11492121320345759, 0.7440633385862897, 0.85876401488240606, 0.80483930692858618,

0.40314698404543869, 1.0535436461406482, 0.87327663221615437,

-0.13934229989014288, 0.72393756926658293, -0.1145312693363762, -1.004095702621026,

-0.83809256709678859, 1.2296322507947566, -0.65244062442795658, -1.7877358794279292,

-1.1881655300730938, 0.10894742036085009, 0.95913292954981111,

0.71393857827320217, -0.50138498577031099, 2.0747735850529407, 1.2814160364125915,

-0.4989921920704794, -0.31866066662602471, -0.51013887802103042, 0.82951757981833529,

0.65559516099093562, -0.87566074115143111, 0.92508545013386267,

8.8609749638116426, -1.7667805818161888, -1.810404696830322, 3.0036018629876162,

1.5811486905867249, -0.20921812054660988, -0.29889238985639327, -1.0267388184256312,

2.7097149747429037, -2.2515282060601285, -2.2404309700198182,

0.26108339564861338, 0.33150318450683769, 0.15091551448064919, 0.72902048902177063,

0.53689374948384716, 0.96083383459606064, 0.059194565387898114, 0.51970721745117976,

0.5786630963836793, 0.98568339651823778, 0.61140705503217085,

0.67408502974761764, 0.20175365515957933, -0.0058315847066409791, 0.78120192972696667,

0.045353035517200914, 1.7371537753252126, -0.52597562547562582, -0.90208297627159129,

0.69312464093465387, -0.30756515832533826, -0.76083772149298801,

4.2948487638860717, -0.20866229540442838, 0.27112036874161588, -0.38846491028523594,

0.81136759636684153, -2.7425928709959937, 1.574524281164815, 0.22674890360374012,

0.13632094407779102, -0.85465213531518736, 0.50799516172323911,

9.6971313547344788, -2.2405429326283155, -2.1802561406935776, 2.0956940604312901,

2.5180111706335588, -0.0037334724458311028, -0.40438408025570699, -1.3188573300971504,

2.5865185205933532, -1.6607829409241781, -1.5306704788271377,

/\* layer 2 \*/

9.4652723154439879, 12.093440164771126, 4.9265075693805578, -1.4412423011222828,

3.5510393773905879, -3.7288162785376344, 3.8472203336952582, 3.4152532147920938,

1.8457963290035171, 2.813468554906799,

-9.0547552742279045, -3.4135629622768024, -1.6123501157265065, 1.1102430622479804,

-0.93531848982606769, 7.3651072358900027, -1.6574514287215174, -1.6036793726589573,

-0.18469990743915979, 7.0307009624562635,

1.0561793702273636, -4.3248094723061898, -0.20802673245701012, 1.9176049664296273,

-0.37609311829409897, -3.0768093781183707, -0.60283450024161656, 0.38072615314690417,

-4.6961331192633384, -2.3746725491819016,

-1.7912711246193735, -4.2247819740812362, -1.6469718282840509, -1.9069215127106751,

-1.4674382430015682, -2.5910079621424313, -1.8083733170901848, -2.1546949672789468,

3.488700668308109, -5.9393699482754743

};

static double listing30Acts[50];

/\* ---------------------------------------------------------- \*/

/\*

listing30Run - run neural network listing30

Input and Output variables.

Variable names are listed below in order, together with each

variable's offset in the data set at the time code was

generated (if the variable is then available).

For nominal variables, the numeric code - class name

conversion is shown indented below the variable name.

To provide nominal inputs, use the corresponding numeric code.

Input variables (Offset):

АТ

Пітливість

1=1

2=2

3=3

Тремор

Мідріаз

1=1

2=2

3=3

Аритмія

1=1

2=2

3=3

Выход:

Стан глікемії

1=D

2=A

3=C

4=B

\*/

/\* ---------------------------------------------------------- \*/

void listing30Run( double inputs[], double outputs[], int outputType )

{

int i, j, k, u;

double \*w = listing30Weights, \*t = listing30Thresholds;

/\* Process inputs - apply pre-processing to each input in turn,

\* storing results in the neuron activations array.

\*/

/\* Input 0: standard numeric pre-processing: linear shift and scale. \*/

if ( inputs[0] == -9999 )

listing30Acts[0] = 0.48926174496644298;

else

listing30Acts[0] = inputs[0] \* 0.01 + -0.29999999999999999;

/\* One-of-N nominal pre-processing \*/

if ( inputs[1] == 0 )

{

double missingValues[] = { 0.32885906040268459, 0.36912751677852351, 0.30201342281879195 };

for ( k=0; k < 3; ++k )

listing30Acts[1+k] = missingValues[k];

}

else

{

for ( k=0; k < 3; ++k )

listing30Acts[1+k] = 0;

listing30Acts[1+(int)inputs[1]-1] = 1;

}

/\* Input 2: standard numeric pre-processing: linear shift and scale. \*/

if ( inputs[2] == -9999 )

listing30Acts[4] = 0.35160594439117926;

else

listing30Acts[4] = inputs[2] \* 0.00071428571428571429 + 0;

/\* One-of-N nominal pre-processing \*/

if ( inputs[3] == 0 )

{

double missingValues[] = { 0.35570469798657717, 0.30201342281879195, 0.34228187919463088 };

for ( k=0; k < 3; ++k )

listing30Acts[5+k] = missingValues[k];

}

else

{

for ( k=0; k < 3; ++k )

listing30Acts[5+k] = 0;

listing30Acts[5+(int)inputs[3]-1] = 1;

}

/\* One-of-N nominal pre-processing \*/

if ( inputs[4] == 0 )

{

double missingValues[] = { 0.33557046979865773, 0.38926174496644295, 0.27516778523489932 };

for ( k=0; k < 3; ++k )

listing30Acts[8+k] = missingValues[k];

}

else

{

for ( k=0; k < 3; ++k )

listing30Acts[8+k] = 0;

listing30Acts[8+(int)inputs[4]-1] = 1;

}

/\*

\* Process layer 1.

\*/

/\* For each unit in turn \*/

for ( u=0; u < 10; ++u )

{

/\*

\* First, calculate post-synaptic potentials, storing

\* these in the listing30Acts array.

\*/

/\* Initialise hidden unit activation to zero \*/

listing30Acts[11+u] = 0.0;

/\* Accumulate weighted sum from inputs \*/

for ( i=0; i < 11; ++i )

listing30Acts[11+u] += \*w++ \* listing30Acts[0+i];

/\* Subtract threshold \*/

listing30Acts[11+u] -= \*t++;

/\* Now apply the hyperbolic activation function, ( e^x - e^-x ) / ( e^x + e^-x ).

\* Deal with overflow and underflow

\*/

if ( listing30Acts[11+u] > 100.0 )

listing30Acts[11+u] = 1.0;

else if ( listing30Acts[11+u] < -100.0 )

listing30Acts[11+u] = -1.0;

else

{

double e1 = exp( listing30Acts[11+u] ), e2 = exp( -listing30Acts[11+u] );

listing30Acts[11+u] = ( e1 - e2 ) / ( e1 + e2 );

}

}

/\*

\* Process layer 2.

\*/

/\* For each unit in turn \*/

for ( u=0; u < 4; ++u )

{

/\*

\* First, calculate post-synaptic potentials, storing

\* these in the listing30Acts array.

\*/

/\* Initialise hidden unit activation to zero \*/

listing30Acts[21+u] = 0.0;

/\* Accumulate weighted sum from inputs \*/

for ( i=0; i < 10; ++i )

listing30Acts[21+u] += \*w++ \* listing30Acts[11+i];

/\* Subtract threshold \*/

listing30Acts[21+u] -= \*t++;

/\* Now calculate exponential of PSP

\*/

if ( listing30Acts[21+u] < -100.0 )

listing30Acts[21+u] = 0.0;

else

listing30Acts[21+u] = exp( listing30Acts[21+u] );

}

/\* Adjust activation to 1.0 sum across layer \*/

{

double sum=0.0;

/\* calculate sum of all activation levels \*/

for ( u=0; u < 4; ++u )

sum += listing30Acts[21+u];

/\* If activations are all near zero, set to equal values.

\* Otherwise, normalise to sum to 1.0 (divide through by sum).

\*/

if ( sum < 1.0e-10 )

{

sum = 1.0 / 4; // (ab)use sum as temporary variable

for ( u=0; u < 4; ++u )

listing30Acts[21+u] = sum;

}

else

{

for ( u=0; u < 4; ++u )

listing30Acts[21+u] /= sum;

}

}

/\* Type of output required - selected by outputType parameter \*/

switch ( outputType )

{

/\* The usual type is to generate the output variables \*/

case 0:

/\* Post-processing, output 0, one-of-N encoding \*/

{

int winner=0;

/\* Find highest output (the "winning" output neuron) \*/

for ( i=1; i < 4; ++i )

if ( listing30Acts[21+i] > listing30Acts[21+winner] )

winner = i;

outputs[0] = winner+1;

}

break;

/\* type 1 is activation of output neurons \*/

case 1:

for ( i=0; i < 4; ++i )

outputs[i] = listing30Acts[21+i];

break;

/\* type 2 is codebook vector of winning node (lowest actn) 1st hidden layer \*/

case 2:

{

int winner=0;

for ( i=1; i < 10; ++i )

if ( listing30Acts[11+i] < listing30Acts[11+winner] )

winner=i;

for ( i=0; i < 11; ++i )

outputs[i] = listing30Weights[11\*winner+i];

}

break;

/\* type 3 indicates winning node (lowest actn) in 1st hidden layer \*/

case 3:

{

int winner=0;

for ( i=1; i < 10; ++i )

if ( listing30Acts[11+i] < listing30Acts[11+winner] )

winner=i;

outputs[0] = winner;

}

break;

}

}

/\*

Test harness. Compile including this main() procedure, as

a windows console program or a DOS program, to interactively

test that the software functions as expected.

\*/

int main(void)

{

int i, outputType=0, noOutputs=1;

double inputs[5], outputs[11];

printf( "\n\nlisting test harness program. Enter inputs below\n" );

printf( "Nominal variables should be numbered starting at 1 (0 for missing)\n" );

printf( "(e.g. if an input is Gender={male,female}, enter 1 for male, 2 for female)\n" );

/\* Infinite (user-breakable) loop for repeated tests \*/

start\_of\_loop:

while ( 1 )

{

/\* Get the input pattern \*/

for ( i=0; i < 5; ++i )

{

printf( "Enter value for input %d: ", i+1 );

scanf( "%lg", & inputs[i] );

/\* Check for sub-menu \*/

if ( inputs[i] == MENUCODE )

{

printf( "Control menu. Select output style, or exit:\n" );

printf( "0. Normal output style (output variable)\n" );

printf( "1. Output layer activations\n" );

printf( "2. Codebook vector (usual only for Kohonen networks\n" );

printf( "3. Winning hidden neuron (ditto only Kohonen)\n" );

printf( "4. Exit program\n" );

printf( "> " );

scanf( "%d", & outputType );

if ( outputType < 0 || outputType > 3 )

{

printf( "\nBye.\n" );

return 0;

}

else

{

/\* Determine how many outputs there are to display (depends on

\* output type)

\*/

switch ( outputType )

{

case 0:

noOutputs = 1;

break;

case 1:

noOutputs = 4;

break;

case 2:

noOutputs = 11;

break;

case 3:

noOutputs = 1;

break;

}

goto start\_of\_loop;

}

}

}

/\* Run the neural network \*/

listing30Run( inputs, outputs, outputType );

/\* Display the output of the neural network \*/

printf( "\n\nOutput of neural network:\n" );

for ( i=0; i < noOutputs; ++i )

{

printf( "Output %d: ", i+1 );

printf( "%g\n", outputs[i] );

}

printf( "\nEnter next input pattern (for control menu inc. exit, enter %d for any input):\n", MENUCODE );

}

return 0;

}