Мета: засвоїти основні підходи до розв’язання задачі класифікації засобами Python.

Теоретичні відомості

Задача класифікації, так само, як і регресія, є прикладом задачі машинного навчання з вчителем.

Метою класифікації є прогнозування мітки класу з заздалегідь визначеної множини можливих значень. Зазвичай, розрізняють бінарну та мультикласову класифікацію. Наприклад, класифікація електронної пошти на спам та не спам є прикладом бінарної класифікації, а прогнозування типу рослин – приклад мультикласової класифікації.

Логістична регресія.

Використовується для побудови регресії, значення якої можуть знаходитись на відрізку [0,1], наприклад, для прогнозування ймовірності. Також використовується для бінарної класифікації. Класифікатор на основі логістичної регресії є лінійним.

Основні рівняння схожі на звичайну регресію, однак для параметричної функції використовується логістична функція (рис. 1):

$$f(x) = \frac{1}{1+e^{x}}.$$

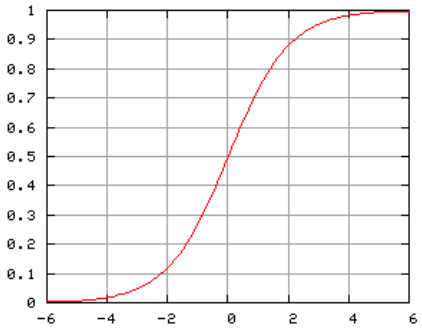

Рис. 1 – Логістична функція

Програмна реалізація:

http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Дерева рішень (Decision trees).

Прикладом дерев ухвалення рішень, можуть бути дерева пошуку, які генеруються алгоритмами пошуку, наприклад, алгоритмом бінарного пошуку. Значення атрибуту порівнюється із значеннями в вершинах дерева і, в залежності від результату порівняння, приймається рішення про подальший шлях порівнянь.

Дерева рішень дозволяють побудувати ієрархію правил «якщо… то» (тести), які приводять до розв'язку задачі. Для задачі класифікації, листами дерев рішень будуть всі можливі класи.

Принцип побудови дерев полягає у розробці системи правил, або поділів вихідного набору даних, таким чином, щоб оптимізувати цільову функцію. Зазвичай, кожне правило є найбільш інформативним, і дозволяє максимально відокремити один клас від інших. На кожному кроці побудови дерева правило, що формується в вузлі, ділить навчальну вибірку на дві частини – частину, в якій виконується правило (нащадки ліворуч) та частина, в якій правило не виконується (нащадки праворуч) (рис. 2).

Алгоритми навчання: ID3, C4.5, CART (Classification and Regression Tree).

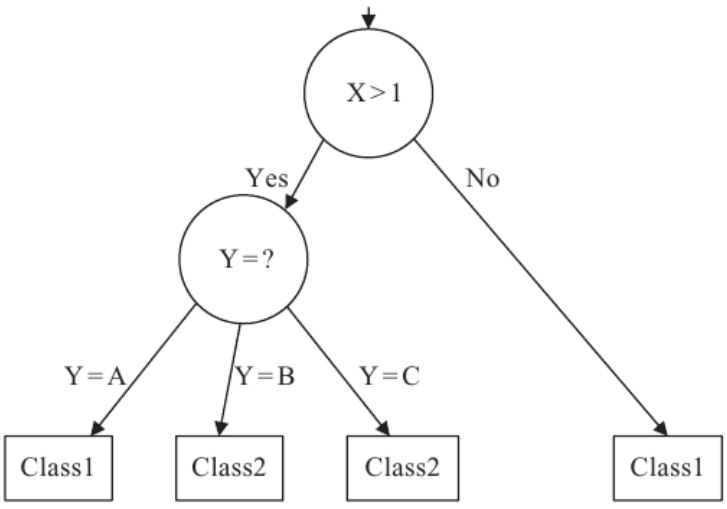

Рис. 2 – Дерево рішень для бінарної класифікації

Як цільова функція застосовується інформаційна ентропія або індекс Gini.

Інформаційна ентропія. Це міра невизначеності системи. Наприклад, для природніх мов, ймовірність появи деяких букв ('a','o','i','e') в слові

вища, ніж інших (’й’,’ь’). Тому невизначеність появи деяких букв нижче, ніж інших. К. Шеннон ввів наступне визначення ентропії в теорії інформації:

$$H(x) = -\sum_{i=1}^{n} p_i \log_2 p_i ,$$

де x - випадкова величина, яка має закон розподілу:

x	x_1	x_2	…	x_i	…	x_n
p	p_1	p_2	…	p_i	…	p_n

Індекс Gini. Міра якості розподілення на класи. Якщо набір даних T містить дані n класів, тоді:

$$Gini(T) = 1 - \sum_{i=1}^{n} p_i^2 ,$$

де p_i - відносна частота класа i в наборі T.

Розглянемо приклад обчислення якості розбиття на класи в певному вузлі з N навчальними прикладами, якщо L та R – кількість навчальних прикладів відповідно в лівому та правому нащадкові; l_i, r_i – число екземплярів i-го класу в лівому\правому нащадкові. Цільова функція якості розбиття на класи в такому випадку:

$$Gini_{split} = \frac{L}{N}\left(1 - \sum_{i=1}^{n}\left(\frac{l_i}{L}\right)^2\right) + \frac{R}{N}\left(1 - \sum_{i=1}^{n}\left(\frac{r_i}{L}\right)^2\right) \to \min .$$

Схема алгоритмів ID3, C4.5.

1. Обрати невикористані ознаки і розрахувати їх ентропію відносно можливих правил.
2. Обрати оптимальну ознаку. Нехай, A – ознака, яка краще за інші класифікує навчальну вибірку (ентропія мінімальна).
3. Створити новий вузол дерева з правилом, в якому використовується ознака A.

Схема алгоритму CART схожа на ID3, C4.5, але як цільова функція використовується індекс Gini. Алгоритм підбирає таку систему правил, яка б мінімізувала індекс якості розбиття на класи Gini.

За допомогою дерев прийняття рішень будуються як лінійні, так і нелінійні класифікатори.

Програмна реалізація:

http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

Ансамблі моделей – це методи, які об'єднують декілька моделей з метою отримати більш точну модель. Найбільш популярні ансамблі дерев рішень – випадковий ліс та градієнтний бустінг.

Випадковий ліс (Random forest). Це набір дерев рішень, в якому кожне дерево дещо відрізняється від іншого.

Основна мета побудови випадкового лісу – зменшити перенавчання окремих дерев. Якщо буде побудовано ансамбль різних дерев, які перенавчаються на різних даних, можна зменшити ступінь перенавчання сукупності дерев за рахунок усереднення результату.

Для побудови випадкового лісу частіше використовуються дві стратегії: навчання дерев на різних частинах навчальної вибірки та використання різної кількості ознак при навчанні окремих дерев.

Програмна реалізація:

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Градієнтний бустінг. На відміну від випадкового лісу, градієнтний бустінг будує послідовність дерев, в якій кожне дерево намагається виправити помилки попереднього. Основна ідея цього підходу полягає у об'єднанні великої кількості простих моделей. Кожне дерево, зазвичай, добре прогнозує частину даних, тому для ітеративного навчання використовується велика кількість дерев.

Програмна реалізація:

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

Метод k-найближчих сусідів (k-Nearest Neighbors). Основна ідея підходу полягає у пошуку схожих прикладів у навчальних даних та об'єднанні їх у один клас. Кожен приклад навчальних даних розглядається як точка в *n*-мірному просторі; кожна з *n* ознак представляє собою координату в просторі. Мірою схожості між прикладами, частіше за всього, є відстань між точками в *n*-мірному просторі. Наприклад, якщо набір даних містить 150 прикладів з 4-х ознак, геометричної інтерпретацією буде 150 точок в чотиривимірному просторі.

При обчисленні класів для тестових прикладів визначається, до якого з класів «тяжіє» даний тестовий приклад. При цьому, обчислюється відстань до *k* найближчих точок (рис. 3).

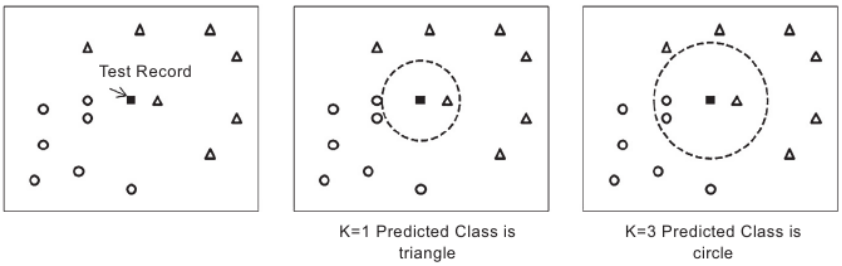

Рис. 3 – Робота метода при різних значення параметру *k*

Програмна реалізація:

http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

Метод опорних векторів (Support Vector Machine).

Один з основних методів класифікації. Метод може використовуватися для побудови нелінійних класифікаторів.

В методі опорних векторів кожна точка в *n*-вимірному просторі представляється як *n*-вимірний вектор, оскільки з точною пов'язано набір її *n* координат. Ідея метода полягає у побудові гіперплощини класифікатора (розмірності *n-1*) найбільш віддаленої від крайніх спостережень (крайніх точок області). Такі точки, власне, і називаються опорними векторами.

Ілюстрація роботи метода у випадку лінійної класифікації наведена на рис. 4. В цьому випадку, кожна точка на площині – це спостереження за двома ознаками. Задача метода SVM – знайти пряму, яка найкращим чином розбиває обидва класи. Опорні вектори обведені.

Найбільша віддаленість гіперплощини класифікатора від опорних векторів призводить до кращого узагальнення моделі. Тобто, прогнози для нових спостережень, які не використовувались при навчанні, будуть більш точними.

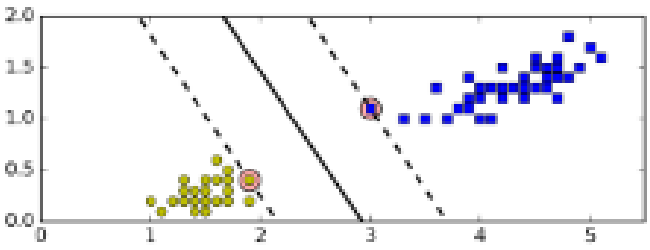

Рис. 4 – Ілюстрація роботи лінійного класифікатора SVM

Програмна реалізація:

http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

Параметри класифікатора бібліотеки scikit-learn впливають на ступінь регуляризації моделі. Рис. 5 ілюструє побудову нелінійного класифікатора при різних параметрах регуляризації.

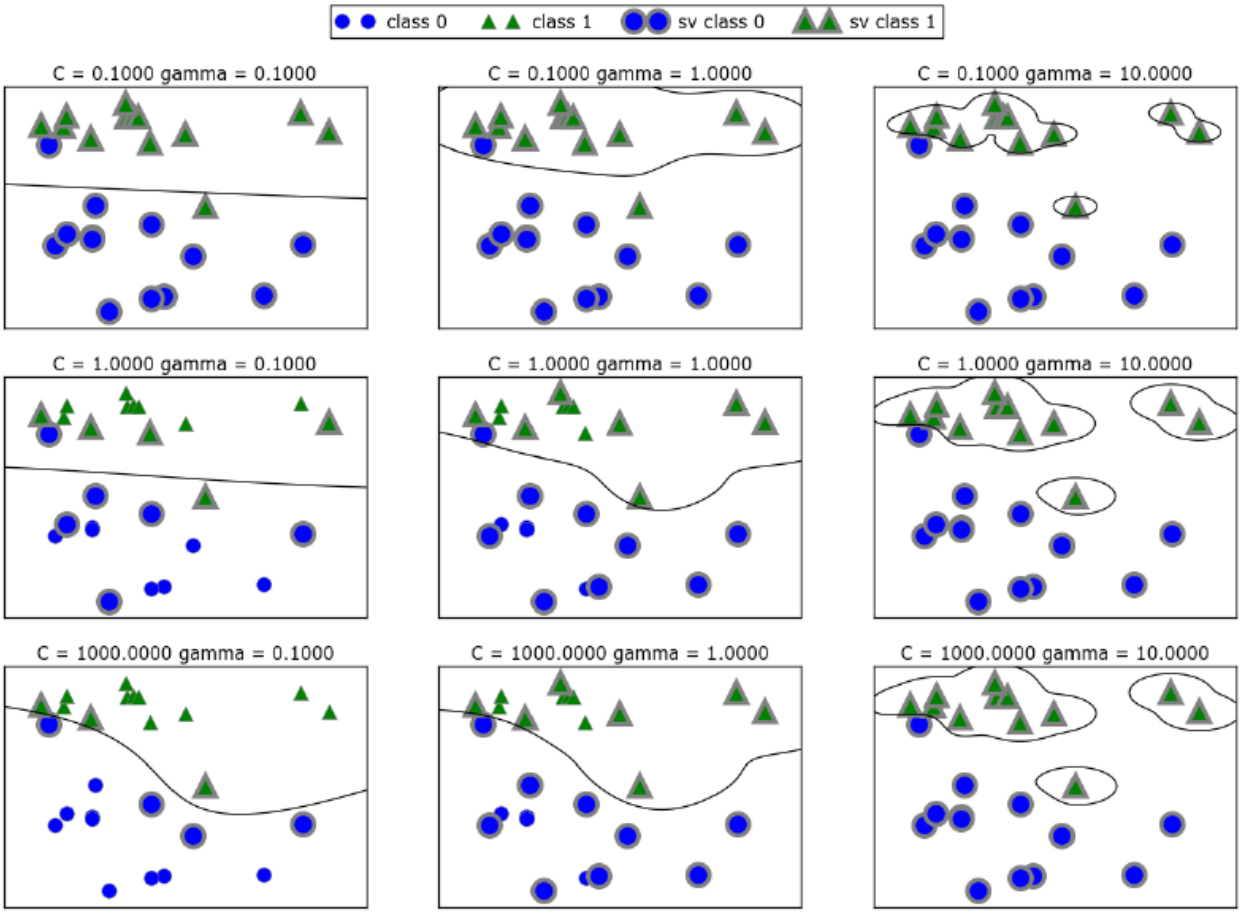

Рис. 5 – Вплив параметрів регуляризації на границю класифікатора

Наївний класифікатор Баєса (Naïve Bayesian).

В основі класифікатора лежить теорема Баєса, яка дозволяє розрахувати умовну ймовірність події A при умові, що відбулася подія B $\left(P\left(A|B\right)\right)$:

$$P\left(A|B\right)=\frac{P\left(B|A\right)P\left(A\right)}{P\left(B\right)},$$

де $P\left(B|A\right)$ – ймовірність події B при умові, що відбулася подія A; $P\left(A\right)$ та $P\left(B\right)$ – ймовірності подій A та B незалежно одна від одної.

Роботу класифікатора Баєса можна розглянути на прикладі. Припустимо, що нам необхідно проаналізувати текстове повідомлення та

визначити його тональність, тобто, віднести його до класу позитивних повідомлень чи негативних. Приналежність до одного з двох класів можна визначити за деякими словами-маркерами, які можуть бути присутніми в повідомленні з негативним чи позитивним забарвленням. Нехай, C – клас повідомлення (позитивне або негативне); F_1 – випадкова подія, в повідомленні зустрічається слово «awesome»; F_2 – випадкова подія, в повідомленні зустрічається слово «crazy». Класифікатор Баєса обчислює ймовірність приналежності повідомлення класу C, якщо відомі ознаки F_1 та F_2 – $P(C \mid F_1, F_2)$. Після застосування теореми Баєса отримаємо формулу:

$$P(C \mid F_1, F_2) = \frac{P(F_1, F_2 \mid C) P(C)}{P(F_1, F_2)},$$

де

$P(C)$ – апріорна ймовірність класу, яка не враховує додаткових знань про дані. Оцінити цю ймовірність можна підрахувавши долю прикладів в кожному класі з навчальної вибірки;

$P(F_1, F_2)$ – ймовірність одночасної появи ознак F_1 та F_2 в повідомленні;

$P(F_1, F_2 \mid C)$ – ймовірність появи ознак F_1 та F_2 в повідомленні, яке належить класу C. Обчислення цього показника досить складне. «Наївність» класифікатора Баєса полягає у припущенні, що одна ознака не впливає на іншу. Це призводить до спрощення розрахункової формули:

$$P(F_1, F_2 \mid C) = P(F_1 \mid C) P(F_2 \mid C).$$

Отже, кінцева формула наївного класифікатора:

$$P(C \mid F_1, F_2) = \frac{P(F_1 \mid C) P(F_2 \mid C) P(C)}{P(F_1, F_2)}.$$

Припустимо, що тривіальна навчальна вибірка для виглядає наступним чином:

Табл. 1 – Приклад навчальної вибірки для класифікатора Баєса

Повідомлення	**Клас**
awesome	позитивний
awesome	позитивний
awesome crazy	позитивний
crazy	позитивний
crazy	негативний

crazy	негативний

Наведемо далі розрахунки алгоритму за наведеними формулами.

Апріорна ймовірність класів:

$$P(C='pos')=\frac{4}{6}\approx 0.67,$$

$$P(C='neg')=\frac{2}{6}\approx 0.33.$$

Ймовірність різних слів в повідомленні в залежності від класу:

$$P(F_1=1|C='pos')=\frac{3}{4},$$

$$P(F_1=0|C='pos')=\frac{1}{4},$$

$$P(F_2=1|C='pos')=\frac{2}{4},$$

$$P(F_1=1|C='neg')=\frac{0}{2},$$

$$P(F_2=1|C='neg')=\frac{2}{2}.$$

Ймовірність появи двох слів в повідомленні, за формулою повної ймовірності:

$$P(F_1,F_2)=P(F_1,F_2|C='pos')P(C='pos')+P(F_1,F_2|C='neg')P(C='neg').$$

$$P(F_1=1,F_2=1)=\frac{3}{4}\cdot\frac{2}{4}\cdot\frac{4}{6}+0\cdot 1\cdot\frac{2}{6}=\frac{1}{4},$$

$$P(F_1=1,F_2=0)=\frac{3}{4}\cdot\frac{2}{4}\cdot\frac{4}{6}+0\cdot 0\cdot\frac{2}{6}=\frac{1}{4},$$

$$P(F_1=0,F_2=1)=\frac{1}{4}\cdot\frac{2}{4}\cdot\frac{4}{6}+\frac{2}{2}\cdot\frac{2}{2}\cdot\frac{2}{6}=\frac{1}{3}.$$

Ймовірність певного класу для нового повідомлення обчислюється:

$$P(C='pos'|F_1,F_2)=\frac{P(F_1|C='pos')P(F_2|C='pos')P(C='pos')}{P(F_1,F_2)},$$

$$P(C='neg' \mid F_1, F_2) = \frac{P(F_1 \mid C='neg')P(F_2 \mid C='neg')P(C='neg')}{P(F_1, F_2)}.$$

Для класифікації необов'язково обчислювати ймовірність за наведеними формулами, достатньо, оцінити, яка з ймовірностей максимальна:

$$C_{best} = \arg\max_{c \in C}\left(P(C=c)P(F_1 \mid C=c)P(F_2 \mid C=c)\right).$$

Програмна реалізація:

http://scikit-learn.org/stable/modules/naive_bayes.html

В scikit-learn реалізовано декілька типів класифікаторів Баєса, основні перераховані нижче.

GaussianNB. Застосовується, коли ознаки числові та розподілені за нормальним законом (гаусіаною). Наприклад, визначення статі за ростом та вагою.

MultinomialNB. Застосовується, коли ознаки є лічильниками деяких подій. Наприклад, при виявленні спаму чи визначенні тональності повідомлення.

BernoulliNB. Застосовується, коли ознаками є бінарні значення «входить до класу», «не входить до класу».

Нейронні мережі.

Налаштування нейронної мережі у випадку класифікації схожі на налаштування з лабораторної роботи №4. Різниця полягає в тому, що кількість значень, які видає вихідний шар дорівнює кількості класів.

Програмна реалізація: http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

Глибинні нейронні мережі.

Багатошаровий перцептрон, який частіше за всього використовується для задачі класифікації може реалізовуватись засобами бібліотеки Keras. При цьому, э можливість задавати довільну кількість прихованих шарів, тобто, будувати глибинну нейронну мережу.

Програмна реалізація: https://keras.io/

Метрики якості класифікації.

Матриця розбіжностей (Confusion matrix). Ефективність прогнозування розробленої моделі за кожним класом можна розглянути за допомогою матриці розбіжностей. Індекси за строками позначають фактичні

класи, а за стовпцями – спрогнозовані. Таким чином, матриця розбіжностей ідеального класифікатора є діагональною.

Програмна реалізація: http://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

ROC-крива (receiver operating characteristic). Відомий метод оцінки точності бінарної класифікації. Якщо існують два класи, які умовно можна назвати позитивний (positive) та негативний (negative), то можна виділити наступні категорії для оцінки точності класифікації (рис. 6):

Істинно-позитивні (true-positive, TP). Спрогнозований клас – позитивний, фактичний клас – позитивний.

Помилково-позитивний (false-positive, FP). Спрогнозований клас – позитивний, фактичний клас – негативний.

Істинно-негативний (true-negative, TN). Спрогнозований клас – негативний, фактичний клас – негативний.

Помилково-негативний (false-negative, FN). Спрогнозований клас – негативний, фактичний клас – позитивний.

Використовуючи описані вище категорії вводять наступні характеристики точності прогнозування.

Частка вірно спрогнозованих класів серед об’єктів, які відносяться до позитивного класу (True positive rate, TPR):

$$TPR = \frac{TP}{TP + FN}.$$

Частка невірно спрогнозованих класів серед об’єктів, які відносяться до негативного класу (False positive rate, FPR):

$$FPR = \frac{FP}{TN + FP}.$$

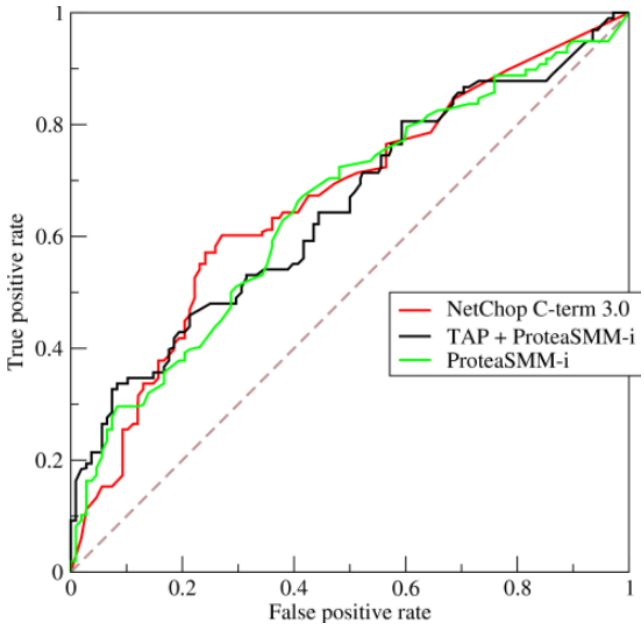

Рис. 6 – Приклад ROC-кривої

Зазвичай, ROC-крива точного класифікатора розташовується вище прямої $y = x$. Це пов'язано з тим, що якісний класифікатор максимізує TPR та мінімізує FPR.

Програмна реалізація: http://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve

Точність класифікації (Accuracy classification score). Для мультикласової класифікації ця функція обчислює точність як частка кількісті спрогнозованих міток, які точно відповідають фактичному набору міток.

Програмна реалізація: http://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html#sklearn.metrics.accuracy_score

Завдання

1. Оберіть для подальшого аналізу один з наведених наборів даних:

- https://www.kaggle.com/uciml/horse-colic
- https://archive.ics.uci.edu/ml/datasets/Activity+Recognition+from+Single+Chest-Mounted+Accelerometer#

2. Використовуйте приклади та розв’язання лабораторних робіт 1–3 для попереднього аналізу та візуалізації даних.
3. Використовуйте приклад 1 на обраному в п.1 наборі даних для побудови наступних класифікаторів:
   - логістичної регресії;
   - дерева рішень;
   - випадковий ліс;
   - k-найближчих сусідів;
   - метод опорних векторів;
   - класифікатор Баєса;
   - нейронні мережі.
4. Проаналізуйте точність побудованих класифікаторів.

Контрольні запитання

1. Загальна постановка задачі класифікації.
2. Методи класифікації.
3. Які метрики використовуються при оцінці якості класифікації?