DOI 10.6084/m9.figshare.22886720
Challenges and Issues of Modern Science, Vol. 1 (2023)

Використання алгоритму машинного навчання Support Vector Machine (SVM) для аналізу даних задля вирішення доцільності обрання професій медичного напрямку

Олексій Миколайович Ізмалков
ORCID: https://orcid.org/0009-0005-3732-7474
Дніпровський національний університет імені Олеся Гончара, Дніпро

Вступ

В теперішній час задля автоматизації машинного навчання необхідно проаналізувати їх переваги та недоліки для автоматизації машинного навчання.

Існує кілька алгоритмів машинного навчання, які можна використовувати для аналізу даних, зібраних з анкети та датчиків, щоб визначити шаблони, які можуть вказувати на придатність людини до різних медичних професій.

Мета та задачі

Визначення найкращого алгоритму для аналізу задля обрання найкращої практики.

Матеріали і методи

Для досягнення даної мети та задачі використовуються можливості Машини опорного вектору (SVM). Вбудовані алгоритми алгоритм можна застосовувати, наприклад, для класифікації людей у різні медичні професії на основі їхніх психофізичних рис.

Random Forest – цей алгоритм можна використовувати для визначення найважливіших психофізичних рис, які вказують на придатність людини до різних медичних професій. Він працює шляхом створення гіперплощини, яка розділяє дані на різні класи шляхом побудови кількох дерев рішень і агрегування їхніх прогнозів.

Такі моделі можна навчити на даних, зібраних із датчиків, щоб передбачити придатність людини до різних медичних професій.Точність прогнозів можна підвищити шляхом оптимізації гіперпараметрів алгоритмів і використання таких методів, як перехресна перевірка, для оцінки їх продуктивності.

SUPPORT VECTOR MACHINE (SVM)

Support Vector Machine (SVM) — це популярний алгоритм машинного навчання, який використовується для класифікації та регресійного аналізу. Це контрольований алгоритм навчання, який можна використовувати для бінарної класифікації, багатокласової класифікації та завдань регресії.

Основною метою SVM є знаходження оптимальної гіперплощини у просторі з високою розмірністю, яка найкращим чином розділяє дані різних класів. Головна ідея полягає в тому, що SVM знаходить найширшу можливу гіперплощину, яка відокремлює класи один від одного, і максимізує відстань (заздалегідь визначений як "проміжок") між цією гіперплощиною та найближчими до неї точками обох класів. Ці точки, які лежать найближче до гіперплощини, називаються опорними векторами.

У SVM кожна точка даних представлена як вектор у просторі великої розмірності, і алгоритм намагається знайти найкращу гіперплощину, яка розділяє точки даних різних класів. Гіперплощина вибирається таким чином, щоб максимально збільшити відстань між найближчими точками даних різних класів. Ці найближчі точки даних називаються опорними векторами, звідси й назва опорна векторна машина.

SVM використовує функцію ядра для перетворення точок даних у простір більшої розмірності, де їх легше розділити. Найпоширенішими ядерними функціями є ядра лінійної, поліноміальної та радіальної базисної функції (RBF).

Коли гіперплощину знайдено, SVM може класифікувати нові точки даних залежно від того, з якого боку гіперплощини вони розташовуються. SVM також можна використовувати для завдань регресії, знаходячи найкраще підходящу гіперплощину, яка мінімізує похибку між прогнозованими та фактичними значеннями.

SVM має кілька переваг, таких як ефективність у просторах великої розмірності, хороша продуктивність узагальнення та здатність обробляти нелінійно розділені дані. Однак SVM може бути чутливим до вибору функцій і параметрів ядра, і це може бути обчислювально дорогим для великих наборів даних.

Загалом SVM — це потужний алгоритм машинного навчання, який можна використовувати для різноманітних завдань класифікації та регресії. Його здатність обробляти багатовимірні та нелінійно розділені дані робить його популярним вибором у багатьох програмах. Розглянемо приклад алгоритму для опорної векторної машини (SVM):

1) Ініціалізуйте модель SVM за допомогою типу ядра (лінійна, поліноміальна, радіальна базисна функція тощо) та параметра регуляризації (C).

2) Розділіть дані на навчальні та тестові набори.

3) Навчіть модель SVM на навчальному наборі за допомогою функції fit().

4) Передбачте мітки класів для тестового набору за допомогою функції predict().

5) Обчисліть точність моделі SVM, порівнявши передбачувані мітки класу з фактичними мітками класу в наборі для тестування.

6) Якщо точність незадовільна, налаштуйте параметри моделі SVM (тип ядра та параметр регулярізації) і повторюйте кроки 3-5, доки не буде досягнуто задовільної точності.

Використовуємо мову програмування Python для реалізації SVM за допомогою бібліотеки scikit-learn:

```
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
# Load the dataX, y = load_data()
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X,
y, test_size=0.2, random_state=42)
# Initialize the SVM model with a linear kernel and
regularization parameter C=1
clf = svm.SVC(kernel='linear', C=1)
# Train the SVM model on the training set
clf.fit(X_train, y_train)
# Predict the class labels for the testing set
y_pred = clf.predict(X_test)
# Calculate the accuracy of the SVM model
accuracy = accuracy_score(y_test, y_pred)
# Print the accuracy
print("Accuracy:", accuracy)
```

Random Forest— це популярний алгоритм машинного навчання, який використовується як для завдань класифікації, так і для регресії. Він належить до категорії методів ансамблевого навчання, де для прогнозування використовуються кілька дерев рішень.

Алгоритм працює шляхом створення лісу дерев рішень, де кожне дерево навчається на випадковій підмножині навчальних даних і випадковій підмножині функцій. Під час процесу навчання алгоритм ітеративно вибирає найкращу точку розділення для кожної функції та створює дерево рішень, яке найкраще відповідає даним.

Щоб зробити прогноз, алгоритм запускає вхідні дані через кожне дерево рішень у лісі та обчислює середнє значення прогнозованих значень з усіх дерев. Це допомагає зменшити переобладнання та підвищити точність моделі.

Однією з ключових переваг Random Forest є те, що він може обробляти велику кількість вхідних функцій і може виявляти взаємодії функцій, які можуть бути пропущені іншими алгоритмами машинного навчання. Він також стійкий до зашумлених даних і може обробляти відсутні значення у вхідних даних.

Random Forest успішно застосовувався в різних сферах, включаючи біоінформатику, фінанси та класифікацію зображень. Однак це може бути дорогим з точки зору обчислень і може потребувати великого обсягу пам'яті для зберігання лісу дерев рішень.

Random Forest — це тип алгоритму машинного навчання, який використовується для завдань класифікації та регресії. Він заснований на концепції дерев рішень, які є діаграмами, які допомагають у прийнятті рішень, розбиваючи проблему на серію запитань «так чи ні». У Random Forest замість одного дерева рішень у нас є ліс дерев рішень. Кожне дерево в лісі вирощується незалежно, використовуючи підмножину навчальних даних і випадкову підмножину ознак. Цей процес допомагає зменшити переобладнання та підвищити точність моделі.

Коли ми хочемо класифікувати нову точку даних, ми вводимо її в усі дерева рішень у лісі та приймаємо більшість голосів, щоб визначити остаточну класифікацію. Такий підхід допомагає підвищити надійність і точність моделі.

Наприклад, передбачення, чи є у пацієнта діабет чи ні, на основі його віку, ІМТ та рівня цукру в крові. Ми можемо навчити модель, використовуючи історичні дані про пацієнтів із відомим статусом діабету. Модель вивчатиме зв'язок між віком, ІМТ, рівнем цукру в крові та ймовірністю діабету. Коли модель навчена, ми можемо використовувати її, щоб передбачити, чи є у нового пацієнта діабет чи ні, на основі його віку, ІМТ та рівня цукру в крові. Модель аналізуватиме дані нового пацієнта та використовуватиме дерева рішень у лісі, щоб зробити прогноз. Потім ми можемо використовувати цю інформацію для надання індивідуальної медичної допомоги пацієнту.

```
python
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
# Load the iris datasetiris = load_iris()
# Split the data into training and testing sets
```

```
X_train, X_test, y_train, y_test =
train_test_split(iris.data, iris.target,
test_size=0.3)
# Initialize the Random Forest classifier with 100
trees
rf = RandomForestClassifier(n_estimators=100)
# Train the classifier on the training data
rf.fit(X_train, y_train)
# Use the classifier to predict the target values for
the testing data
y_pred = rf.predict(X_test)
# Print the accuracy score of the classifier on the
testing data
print("Accuracy:", rf.score(X_test, y_test))
```

У цьому прикладі ми спочатку завантажуємо набір даних і розділяємо його на навчальний і тестовий набори. Потім ми ініціалізуємо класифікатор випадкового лісу зі 100 деревами та навчаємо його на навчальних даних. Нарешті, ми використовуємо класифікатор для прогнозування цільових значень для даних тестування та друкуємо оцінку точності класифікатора на даних тестування. Цей класифікатор дозволяє визначити ймовірність та похибку вимірювання що надає повний спектр дослідження та аналіз задачі та дослідження.

Висновок

Ці алгоритми можна навчити на даних, зібраних з анкети та датчиків, щоб передбачити придатність людини до різних медичних професій. Точність прогнозів можна підвищити шляхом оптимізації гіперпараметрів алгоритмів і використання таких методів, як перехресна перевірка, для оцінки їх продуктивності. Для подальшого дослідження будуть використовуватися нові алгоритми та формули задля забезпечення більш точного аналізу даних та аналізу інших дослідження в рамках наукової роботи.

Посилання

1. "Support Vector Machines for Pattern Classification" by Shigeo Abe and Yoichi Takenaka

2. "Support Vector Machines: Theory and Applications" by S. Sathiya Keerthi and C. J. Lin: This book offers a detailed treatment of SVMs, exploring both the theoretical foundations and practical aspects.