Zajęcie 1. Praktyczne zastosowanie regresji liniowej w analizie danych. Implementacja algorytmów klasyfikacji binarnej w Pythonie

Abstract

Celem jest nabycie podstawowej znajomości przepływu pracy uczenia maszynowego rozwiązując zadanie budowania modeli regresji lub klasyfikacji odpowiednio do określonego wariantu

1. Regresja Liniowa Wielowymiarowa - Opis Matematyczny

Regresja liniowa wielowymiarowa to metoda modelowania zależności między zmienną zależną y a wieloma zmiennymi niezależnymi $x_1, x_2, \ldots, x_p$. W przypadku regresji wielowymiarowej równanie modelu przyjmuje postać:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \epsilon_i, \quad i = 1, 2, \ldots, n$$

gdzie:

- y_i to wartość zmiennej zależnej dla obserwacji i,
- $x_{i1}, x_{i2}, \ldots, x_{ip}$ to wartości zmiennych niezależnych dla obserwacji i,
- β_0 to wyraz wolny (przecięcie z osią y),
- $\beta_1, \beta_2, \ldots, \beta_p$ to współczynniki regresji, które modelują wpływ zmiennych $x_1, x_2, \ldots, x_p$ na zmienną zależną y,
- ϵ_i to składnik losowy dla obserwacji i, reprezentujący błąd modelu.

Model ten można również zapisać w bardziej zwięzłej formie macierzowej:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

gdzie:

- $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$ to wektor zmiennych zależnych o rozmiarze $n \times 1$,
- $\mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1p} \\ 1 & x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{np} \end{bmatrix}$ to macierz projektowania o wymiarach $n \times (p+1)$, gdzie każda kolumna odpowiada jednej zmiennej niezależnej, a pierwsza kolumna to wektor jedynek odpowiadający wyrazowi wolnemu β_0,
- $\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix}$ to wektor współczynników regresji o wymiarze $(p+1) \times 1$,
- $\boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$ to wektor błędów o rozmiarze $n \times 1$.

1.1. Oszacowanie Współczynników

Współczynniki regresji $\boldsymbol{\beta}$ są szacowane poprzez minimalizację funkcji kosztu, jaką jest suma kwadratów reszt (błędów predykcji). Funkcja ta jest dana wzorem:

$$J(\boldsymbol{\beta}) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}))^2$$

Aby znaleźć optymalne wartości współczynników $\boldsymbol{\beta}$, minimalizujemy funkcję $J(\boldsymbol{\beta})$. W rozwiązaniu analitycznym używa się równań normalnych:

$$\boldsymbol{\beta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

To wyrażenie zapewnia optymalne wartości współczynników regresji, o ile macierz $\mathbf{X}^T\mathbf{X}$ jest odwracalna.

1.2. Podsumowanie

Regresja liniowa wielowymiarowa jest potężnym narzędziem statystycznym, które pozwala na modelowanie zależności między wieloma zmiennymi niezależnymi a jedną zmienną zależną. Metoda minimalizowania sumy kwadratów reszt pozwala na oszacowanie współczynników modelu, co umożliwia dokonywanie predykcji na podstawie nowych danych.

2. Klasyfikacja Binarna za pomocą Maszyn Wektorów Nośnych (SVM) - Opis Matematyczny

Maszyny wektorów nośnych (ang. Support Vector Machines, SVM) są metodą stosowaną do rozwiązywania problemów klasyfikacji binarnej. Celem SVM jest znalezienie hiperpłaszczyzny, która maksymalizuje margines oddzielający dwie klasy danych.

2.1. Dane i Model

Zakładamy, że mamy zbiór treningowy $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, gdzie:

- $\mathbf{x}_i \in \mathbb{R}^p$ to wektor cech dla i-tej próbki, o wymiarze p,
- $y_i \in \{-1, 1\}$ to etykieta klasy dla i-tej próbki (dwie klasy: -1 i 1).

Celem SVM jest znalezienie hiperpłaszczyzny, która oddziela próbki obu klas, minimalizując błąd klasyfikacji oraz maksymalizując margines. Hiperpłaszczyzna ma postać równania:

$$\mathbf{w}^T\mathbf{x} + b = 0$$

gdzie:

- $\mathbf{w} \in \mathbb{R}^p$ to wektor wag definiujący orientację hiperpłaszczyzny,
- $b \in \mathbb{R}$ to wyraz wolny (przesunięcie hiperpłaszczyzny).

2.2. Maksymalizacja Marginesu

Margines to odległość między hiperpłaszczyzną a najbliższymi punktami treningowymi (zwanymi wektorami nośnymi). Maksymalizacja marginesu prowadzi do lepszego uogólnienia modelu. Odległość od hiperpłaszczyzny dla próbki $\mathbf{x}_i$ wynosi:

$$\frac{|\mathbf{w}^T \mathbf{x}_i + b|}{\|\mathbf{w}\|}$$

Aby maksymalizować margines, musimy rozwiązać następujący problem optymalizacji:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$

przy ograniczeniach zapewniających poprawną klasyfikację:

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 \quad \text{dla wszystkich} \quad i = 1, 2, \ldots, n$$

Problem ten można rozwiązać za pomocą optymalizacji kwadratowej, wprowadzając mnożniki Lagrange'a.

2.3. Problem Dualny

Aby ułatwić rozwiązanie, można zapisać problem w postaci dualnej, gdzie zamiast wektora $\mathbf{w}$ szukamy mnożników α_i. Problem dualny ma postać:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

przy ograniczeniach:

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \quad \text{oraz} \quad 0 \leq \alpha_i \leq C \quad \text{dla wszystkich} \quad i = 1, 2, \ldots, n$$

gdzie C jest hiperparametrem, który kontroluje kompromis między maksymalizacją marginesu a minimalizacją błędu klasyfikacji.

2.4. Funkcja Decyzyjna

Po rozwiązaniu problemu dualnego i znalezieniu optymalnych wartości α_i, współczynniki $\mathbf{w}$ można obliczyć jako:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

Następnie możemy użyć funkcji decyzyjnej, aby klasyfikować nowe próbki $\mathbf{x}$:

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$$

Próbka $\mathbf{x}$ zostaje przypisana do klasy 1, jeśli $f(\mathbf{x}) > 0$, lub do klasy -1, jeśli $f(\mathbf{x}) < 0$.

Jądra (Kernels)

Jeśli dane są nieliniowo separowalne, możemy zastosować funkcję jądra $K(\mathbf{x}_i, \mathbf{x}_j)$, aby przekształcić dane do wyższej przestrzeni wymiarowej. W problemie dualnym zamiast obliczać iloczyn skalarny $\mathbf{x}_i^T \mathbf{x}_j$, stosujemy funkcję jądra:

$$K(\mathbf{x}_i, \mathbf{x}_j)$$

Najczęściej stosowane funkcje jądra to:

- Jądro liniowe: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
- Jądro wielomianowe: $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + 1)^d$
- Jądro RBF (Gaussian): $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2)$

W przypadku zastosowania jąder funkcja decyzyjna przyjmuje postać:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

3. Przepływ pracy uczenia maszynowego

Proponowane podejście zawiera kroki:
1. Definicja problemu
2. Dane
3. Ocena
4. Cechy
5. Modelowanie
6. Eksperymenty

Przykładowa implementacja na przykładzie prognozowania chorób serca jest przedstawiona w tutorialu .pynb. Dodaje się również zbiór danych .csv

4. Tworzenie przepływu pracy uczenia maszynowego w narzędziu KNIME

KNIME (Konstanz Information Miner) jest otwartoźródłowym narzędziem typu platforma do analizy danych, które umożliwia tworzenie przepływów pracy (ang. workflows) do analizy danych, w tym do zastosowań związanych z uczeniem maszynowym. Dzięki graficznemu interfejsowi użytkownika, KNIME pozwala na łatwe tworzenie, testowanie i wdrażanie modeli uczenia maszynowego bez konieczności pisania kodu.

4.1. Tworzenie przepływu pracy

Przepływ pracy w KNIME składa się z serii węzłów (ang. nodes), które są połączone w sposób reprezentujący przepływ danych. Poniżej przedstawione są podstawowe kroki tworzenia przepływu pracy uczenia maszynowego.

4.2. 1. Import danych

Pierwszym krokiem jest załadowanie danych, które będą analizowane i przetwarzane. KNIME wspiera różnorodne źródła danych, w tym pliki CSV, Excel, bazy danych SQL oraz API webowe. Aby zaimportować dane, należy:

- Użyć węzła `File Reader` lub `CSV Reader`, aby załadować dane z plików tekstowych lub CSV.
- Użyć węzła `Excel Reader` do importu danych z plików Excel.
- W przypadku baz danych użyć węzła `Database Reader`, który umożliwia połączenie z bazą danych i import danych za pomocą zapytań SQL.

4.3. 2. Przygotowanie danych

Dane, które są zaimportowane, mogą wymagać przetworzenia przed zastosowaniem modelu uczenia maszynowego. KNIME dostarcza szereg węzłów do przetwarzania danych, takich jak:

- `Column Filter` - umożliwia wybór lub usunięcie niepotrzebnych kolumn z danych.
- `Row Filter` - służy do filtrowania rekordów (wierszy) na podstawie zadanych warunków.
- `Missing Value` - pozwala na obsługę brakujących danych, np. poprzez wypełnianie brakujących wartości średnią, medianą lub wartością domyślną.
- `Normalization` - umożliwia normalizację danych, aby każda cecha miała porównywalną skalę.

4.4. 3. Podział na zbiór treningowy i testowy

W celu oceny jakości modelu uczenia maszynowego, dane są zazwyczaj dzielone na zbiór treningowy oraz testowy. Można to zrobić przy użyciu węzła `Partitioning`, który umożliwia podział danych według zadanego stosunku, np. 70% danych do treningu i 30% do testu.

4.5. 4. Budowa modelu

KNIME wspiera szeroką gamę algorytmów uczenia maszynowego, takich jak regresja liniowa, drzewa decyzyjne, lasy losowe (ang. Random Forest), czy maszyny wektorów nośnych (ang. Support Vector Machines). Aby zbudować model, należy:

- Wybrać odpowiedni węzeł modelujący, np. `Logistic Regression Learner`, `Decision Tree Learner`, `Random Forest Learner` lub `SVM Learner`.
- Połączyć węzeł modelujący z danymi treningowymi.
- Węzeł ten trenuje model na podstawie danych wejściowych.

4.6. 5. Ocena modelu

Po zbudowaniu modelu, należy ocenić jego skuteczność na podstawie danych testowych. KNIME oferuje szereg węzłów do ewaluacji modeli:

- `Scorer` - umożliwia obliczenie miar jakości klasyfikacji, takich jak dokładność, precyzja, czułość, specyficzność i macierz pomyłek.
- `ROC Curve` - generuje krzywą ROC (ang. Receiver Operating Characteristic) oraz oblicza pole pod krzywą (AUC).
- `Numeric Scorer` - w przypadku modeli regresji umożliwia obliczenie takich miar jak średni błąd kwadratowy (MSE) czy współczynnik determinacji R^2.

4.7. 6. Walidacja krzyżowa

Aby uzyskać bardziej wiarygodną ocenę modelu, można zastosować walidację krzyżową (ang. cross-validation). W KNIME można to osiągnąć przy użyciu węzła `X-Partitioner` oraz `X-Aggregator`, które odpowiednio dzielą dane na podzbiory oraz zbierają wyniki z różnych iteracji walidacji.

4.8. 7. Wdrażanie modelu

Po wytrenowaniu i ocenie modelu, KNIME umożliwia wdrożenie modelu do użytku produkcyjnego. Może to obejmować:

- Użycie węzła `PMML Writer` do zapisania modelu w formacie PMML, który może być używany w różnych systemach.
- Zastosowanie węzła `Model Writer`, aby zapisać model do późniejszego użycia w KNIME.
- Integrację z narzędziami do API REST, aby udostępniać model przez interfejsy webowe.

4.9. Przykład przepływu pracy

Na powyższym rysunku pokazano przykładowy przepływ pracy w KNIME, który obejmuje wczytanie danych, ich przetwarzanie, trenowanie modelu uczenia maszynowego oraz jego ocenę.

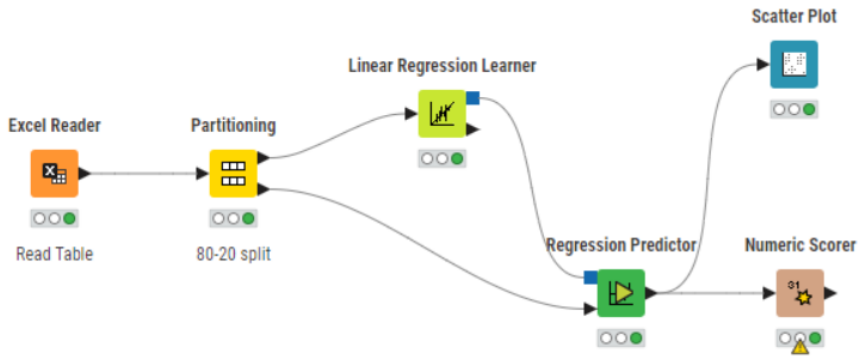

Figure 1: Przykład przepływu pracy w KNIME dla zadania klasyfikacji.

4.10. Podsumowanie

KNIME jest potężnym narzędziem do analizy danych i uczenia maszynowego, które pozwala na szybkie prototypowanie modeli i ich ocenę. Dzięki graficznemu interfejsowi oraz szerokiej bibliotece węzłów, KNIME umożliwia tworzenie zaawansowanych przepływów pracy bez konieczności programowania.

5. Zadanie

Opracować przepływ pracy uczenia maszynowego zagadnienia regresji (model regresji liniowej) oraz klasyfikacji binarnej (model SVM) na podstawie zbioru danych według wariantu zadania:

1. klasyfikacja choroby Parkinsona `https://www.kaggle.com/dipayanbiswas/parkinsons-disease-speech-signal-features?select=pd_speech_features.csv`

2. smoking patients `https://www.kaggle.com/thomaskonstantin/cpg-values-of-smoking-`

3. Powikłania zawału mięśnia sercowego `https://www.kaggle.com/rafatashrafjoy/myocardial-infarction-complications`

4. Sygnały kardiotokografii `https://www.kaggle.com/sohelranaccselab/biomedical-cardiotocography`

5. Badania pH https://www.kaggle.com/zfturbo/measurements-of-urine-ph

6. Analiza cukrzycy https://www.kaggle.com/veerukhannan/diabetes

7. Choroba Alzheimera https://www.kaggle.com/madhucharan/alzheimersdisease5classda

8. Prostate cancer https://www.kaggle.com/ashrafalsinglawi/prostate-cancer-survival

9. klasyfikacja choroby Parkinsona https://www.kaggle.com/dipayanbiswas/parkinsons-disease-speech-signal-features?select=pd_speech_features.csv

10. smoking patients https://www.kaggle.com/thomaskonstantin/cpg-values-of-smoking-

11. Powikłania zawału mięśnia sercowego https://www.kaggle.com/rafatashrafjoy/myocardial-infarction-complications

12. Sygnały kardiotokografii https://www.kaggle.com/sohelranaccselab/biomedical-cardiotocography

13. Badania pH https://www.kaggle.com/zfturbo/measurements-of-urine-ph

14. Analiza cukrzycy https://www.kaggle.com/veerukhannan/diabetes

15. Choroba Alzheimera https://www.kaggle.com/madhucharan/alzheimersdisease5classda

16. Prostate cancer https://www.kaggle.com/ashrafalsinglawi/prostate-cancer-survival

Rozwiązywanie zadania opracować zarówno w KNIME jak Jupyter Notebook.

References

[1] Drew Conway, John Myles White, Uczenie maszynowe dla programistów, Helion, 2015

[2] Beginner's Guide https://www.datacamp.com/community/tutorials/r-packages-guide

[3] Przykładowe rozwiązania http://zasoby.open.agh.edu.pl/~15spkiepas/index.php/przykladowe-rozwiazania/index.html