Helion

<packt>

ZAAWANSOWANE TECHNIKI PRZETWARZANIA JĘZYKA NATURALNEGO

Od podstaw do modeli LLM
i zastosowań biznesowych w Pythonie

LIOR GAZIT, MEYSAM GHAFFARI

Słowo wstępne: **Asha Saxena**, przedsiębiorczyni, profesorka i stratežka AI

Tytuł oryginału: Mastering NLP from Foundations to LLMs: Apply advanced rule-based techniques to LLMs and solve real-world business problems using Python

Tłumaczenie: Grzegorz Werner

ISBN: 978-83-289-2048-4

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
https://helion.pl/user/opinie/zatepr
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *https://helion.pl* (księgarnia internetowa, katalog książek)

Printed in Poland.

Spis treści

ROZDZIAŁ 7

Duże modele językowe — teoria, projektowanie i implementacja 189

Rozdział 3

Wykorzystanie potencjału uczenia maszynowego w NLP

W tym rozdziale poznasz podstawy **uczenia maszynowego** (ang. *machine learning*, **ML**) i technik przetwarzania wstępnego, które mają kluczowe znaczenie w **przetwarzaniu języka naturalnego** (ang. *natural language processing*, **NLP**). ML to potężne narzędzie do budowania modeli, które mogą uczyć się na podstawie danych, a NLP jest jednym z najbardziej ekscytujących i złożonych zastosowań ML.

Pod koniec tego rozdziału będziesz znać tajniki eksploracji danych, przetwarzania wstępnego i dzielenia danych, nauczysz się radzić sobie z niezrównoważonymi danymi i poznasz popularne modele ML, zwłaszcza te, których używa się w kontekście NLP.

W niniejszym rozdziale omówimy następujące zagadnienia:

- Eksploracja danych.
- Popularne modele ML.
- Niedostateczne i nadmierne dopasowanie modelu.
- Dzielenie danych.
- Dostrajanie hiperparametrów.
- Modele zespołowe.
- Dane niezrównoważone.
- Dane skorelowane.

Wymagania techniczne

W tym i kolejnych rozdziałach książki zakładamy, że znasz już języki programowania, szczególnie Pythona. Oczekujemy też, że przeczytałeś poprzednie rozdziały i zapoznałeś się z kluczowymi koncepcjami algebry liniowej i statystyki.

Eksploracja danych

Kiedy pracujesz w środowisku metodologicznym, zbiory danych są często dobrze znane i wstępnie przetworzone; przykładem mogą być zbiory danych Kaggle. Jednak w rzeczywistym środowisku biznesowym ważnym zadaniem jest zdefiniowanie zbioru danych na podstawie wszystkich dostępnych źródeł danych, zbadanie zgromadzonych danych w celu określenia najlepszej metody ich wstępnego przetworzenia, a wreszcie wybranie modeli ML i NLP, które najlepiej pasują do problemu i danych. Proces ten wymaga uważnego rozważenia i przeanalizowania danych, a także dobrego zrozumienia problemu biznesowego.

W NLP dane bywają bardzo złożone, ponieważ często obejmują dane tekstowe i dźwiękowe, które mogą być nieustrukturyzowane i trudne w analizie. Złożoność ta sprawia, że przetwarzanie wstępne jest kluczowym etapem przygotowywania danych na użytek modeli ML. Rozwiązywanie każdego problemu NLP lub ML zaczyna się od eksploracji danych, która pozwala lepiej je zrozumieć i wybrać najlepszy model.

Po wstępnym przetworzeniu danych kolejnym etapem jest zbadanie ich w celu lepszego zrozumienia ich charakterystyki i struktury. Eksploracja danych to proces iteracyjny, które polega na wizualizowaniu i analizowaniu danych, szukaniu wzorców i relacji oraz identyfikowaniu potencjalnych problemów lub elementów odstających. Proces ten pomaga ustalić, które cechy są najważniejsze dla modeli ML, oraz odkryć potencjalną stronniczość albo problemy z jakością danych. Aby zoptymalizować dane i ułatwić modelom ML ich analizowanie, można zastosować takie metody przetwarzania wstępnego jak tokenizacja, tematyzacja i lematyzacja. W tym rozdziale przedstawimy przegląd ogólnych technik przetwarzania wstępnego na potrzeby ML. W kolejnym rozdziale skupimy się na technikach przetwarzania wstępnego specyficznych dla przetwarzania tekstu. Warto podkreślić, że zastosowanie efektywnych technik przetwarzania wstępnego może znacznie zwiększyć wydajność i dokładność modeli ML, żeby działały bardziej niezawodnie.

Wreszcie po wstępnym przetworzeniu i zbadaniu danych możemy przystąpić do budowania modeli ML. Nie ma jednego magicznego rozwiązania, które działałoby we wszystkich problemach ML, więc trzeba dobrze zastanowić się, które modele są najbardziej odpowiednie dla konkretnego problemu i dostępnych danych. Istnieją różne typy modeli NLP, w tym oparte na regułach, statystyczne i wykorzystujące uczenie głębokie. Każdy typ modelu ma swoje wady i zalety, co podkreśla znaczenie wyboru najodpowiedniejszego modelu dla konkretnego problemu i zbioru danych.

Eksploracja danych to ważny początkowy etap w procesie ML, który polega na analizowaniu danych przed zbudowaniem modelu ML. Celem eksploracji danych jest zrozumienie danych, zidentyfikowanie wzorców, wykrycie anomalii i przygotowanie danych do modelowania. Eksploracja pomaga wybrać właściwy algorytm ML i ustalić najlepszy zestaw cech.

Oto kilka technik często używanych w eksploracji danych:

- **Wizualizacja danych**. Wizualizacja danych polega na przedstawianiu danych w formacie graficznym lub obrazkowym. Umożliwia wzrokową eksplorację danych, zapewniając wgląd w ich rozkład, wzorce i relacje. W wizualizacji danych powszechnie używa się takich narzędzi jak wykresy punktowe, wykresy słupkowe, mapy cieplne, wykresy skrzynkowe i macierze korelacji.

- **Oczyszczanie danych**. Oczyszczanie danych to etap przetwarzania wstępnego, w którym identyfikujemy błędy, niespójności oraz brakujące wartości i próbujemy je poprawić. Wpływa ono na ostateczne wyniki modelu, ponieważ modele ML są wrażliwe na błędy w danych. Do najczęściej używanych technik oczyszczania danych należą usuwanie duplikatów i uzupełnianie brakujących wartości.
- **Inżynieria cech**. Inżynieria cech, która polega na tworzeniu nowych cech z istniejących danych, odgrywa kluczową rolę w optymalizowaniu efektywności modeli uczenia maszynowego. Proces ten obejmuje nie tylko identyfikowanie istotnych cech, ale również ich przekształcanie i wprowadzanie nowych. Różne techniki inżynierii cech, w tym skalowanie, normalizacja, ograniczanie wymiarowości i selekcja cech przyczyniają się do poprawy ogólnej trafności modeli.
- **Analiza statystyczna**. W analizie statystycznej wykorzystuje się szeroką gamę technik statystycznych do zbadania danych i zyskania wglądu w ich własności. Do kluczowych metod statystycznych należą testowanie hipotez, analiza regresyjna i analiza szeregów czasowych; wszystkie one pomagają lepiej zrozumieć charakterystykę danych.
- **Wiedza dziedzinowa**. Wykorzystanie wiedzy dziedzinowej obejmuje stosowanie uprzedniej wiedzy o dziedzinie danych w celu dokonywania spostrzeżeń i podejmowania racjonalnych decyzji. Wiedza ta przydaje się do identyfikowania istotnych cech, interpretowania wyników oraz wybierania algorytmu ML, który pasuje najlepiej do określonego zadania.

W kolejnych punktach omówimy każdą z tych technik.

Wizualizacja danych

Wizualizacja danych jest kluczowym komponentem uczenia maszynowego, ponieważ pomaga w zrozumieniu i eksploracji złożonych zbiorów danych. Polega na tworzeniu wizualnych reprezentacji danych z wykorzystaniem diagramów, wykresów i innych pomocy wizualnych. Taki sposób prezentacji danych pomaga odkryć wzorce, trendy i relacje, które nie zawsze są oczywiste podczas badania surowych danych.

W zadaniach NLP wizualizacja danych pomaga zyskać wgląd w lingwistyczne wzorce i struktury w danych tekstowych. Możemy na przykład tworzyć chmury słów, aby zwizualizować częstość występowania słów w korpusie, albo używać map cieplnych, aby przedstawić współwystępowanie wyrazów lub fraz. Możemy też używać wykresów punktowych i liniowych, aby zwizualizować zmianę odczuć albo tematów na przestrzeni czasu.

Popularnym typem wizualizacji w ML jest wykres punktowy, którego używa się do pokazania związków między dwiema zmiennymi. Wykreślając wartości dwóch zmiennych na osiach X i Y, możemy odkryć ich wzajemne relacje albo trendy. Wykresy punktowe szczególnie przydają się do identyfikowania klastrów, czyli grup punktów danych, które mają podobne właściwości.

Innym często stosowanym w ML typem wizualizacji jest histogram, narzędzie, które ilustruje rozkład jednej zmiennej. Grupując dane w przedziałach i przedstawiając częstość

występowania punktów danych w każdym przedziale, możemy odkryć zakres wartości, które dominują w zbiorze danych. Histogramy przydają się do wykrywania elementów odstających lub anomalii i pomagają w rozpoznawaniu obszarów, w których dane cechują się skośnością albo stronniczością.

Oprócz tych podstawowych wizualizacji praktycy ML często używają bardziej zaawansowanych technik, takich jak ograniczanie wymiarowości i wizualizacje sieciowe. Techniki ograniczania wymiarowości, na przykład **analiza głównych składowych** (ang. *principal component analysis*, **PCA**) albo **stochastyczne osadzanie sąsiadów w oparciu o rozkład *t*** (ang. *t-distributed stochastic neighbor embedding*, ***t*-SNE**), zmniejszają liczbę wymiarów i ułatwiają wizualizowanie lub analizowanie danych. Wizualizacje sieciowe służą natomiast do przedstawiania złożonych relacji między danymi, takich jak współwystępowanie słów albo połączenia między użytkownikami serwisu społecznościowego.

Oczyszczanie danych

Oczyszczanie danych polega na rozpoznawaniu i poprawianiu błędów, niespójności i niedokładności w zbiorze danych. Ta kluczowa faza przygotowywania danych na użytek ML znacznie zwiększa dokładność i trafność modelu, które w dużej mierze zależą od jakości danych użytych do treningu. Używa się do tego wielu standardowych technik. Przyjrzyjmy się im bliżej.

Brakujące wartości

Brakujące dane to problem, który pojawia się w wielu projektach uczenia maszynowego. Obsługa brakujących wartości jest ważna, ponieważ modele ML nie radzą sobie z brakiem danych i albo zgłaszają błędy, albo generują niedokładne wyniki.

Istnieje kilka metod radzenia sobie z brakującymi danymi w projektach ML:

- **Odrzucanie wierszy**. Najprostszym podejściem jest odrzucanie wierszy, które nie zawierają potrzebnych wartości. Metoda ta wymaga jednak ostrożności, ponieważ usunięcie nadmiernej liczby wierszy może doprowadzić do utraty cennych danych i niekorzystnie wpłynąć na ogólną dokładność modelu. Zwykle używa się jej, kiedy zbiór danych zawiera wiele wierszy, a tylko w kilku brakuje wymaganych wartości. W takim przypadku usunięcie kilku wierszy pozwala łatwo wytrenować model bez większego wpływu na jego ostateczne działanie.
- **Odrzucanie kolumn**. Innym podejściem jest odrzucenie kolumn, w których brakuje wartości. Metoda ta może być skuteczna, jeśli brakujące wartości są skupione w nielicznych kolumnach, a kolumny te nie są istotne dla analizy. Jednakże odrzucenie ważnych kolumn może prowadzić do utraty wartościowych informacji. Przed ich odrzuceniem warto przeprowadzić jakiś rodzaj analizy korelacyjnej, aby określić korelację wartości w tych kolumnach z klasą lub wartością docelową.
- **Imputacja średniej/mediany/mody**. Imputacja średniej/mediany/mody polega na zastępowaniu brakujących wartości średnią, medianą lub modą (dominantą) obliczoną na podstawie istniejących wartości w odpowiedniej kolumnie. Metoda ta jest łatwa w implementacji i bywa skuteczna, jeśli

brakujące wartości są nieliczne i rozłożone losowo, ale może również wprowadzać stronniczość i wpływać na zmienność danych.

- **Imputacja regresyjna**. Imputacja regresyjna polega na przewidywaniu brakujących wartości na podstawie wartości innych zmiennych w zbiorze danych. Metoda ta bywa efektywna, jeśli brakujące wartości są związane z innymi zmiennymi w zbiorze danych, ale wymaga zbudowania modelu regresji dla każdej kolumny, w której brakuje wartości.
- **Imputacja wielokrotna**. Imputacja wielokrotna obejmuje generowanie wielu imputowanych zbiorów danych z użyciem modeli statystycznych, a następnie łączenie wyników w celu utworzenia miarodajnego zbioru danych. Podejście to okazuje się skuteczne, zwłaszcza jeśli mamy do czynienia z rozłożonymi nielosowo brakującymi wartościami i dużą liczbą luk w zbiorze danych.
- **Imputacja metodą k najbliższych sąsiadów**. Metoda ta polega na identyfikowaniu k punktów danych najbliższych brakującej wartości i określaniu jej na podstawie ich wartości. Metoda ta jest skuteczna, jeśli brakujące wartości należą do klastrów w zbiorze danych. Polega ona na znajdowaniu rekordów najbardziej podobnych do tego, w którym brakuje wartości, i używaniu średniej wartości tych rekordów do uzupełnienia brakującej wartości w danym rekordzie.

Zasadniczo wybór metody obsługi brakujących wartości zależy od takich czynników, jak natura i zakres braków, cele analizy i dostępność zasobów. Trzeba uważnie przemyśleć wady i zalety każdej metody i wybrać tę, która sprawdzi się najlepiej w danym projekcie.

Usuwanie duplikatów

Eliminowanie duplikatów to etap przetwarzania wstępnego, który pozwala oczyścić zbiory danych poprzez wykrywanie i usuwanie identycznych rekordów. Występowanie zduplikowanych rekordów może wynikać z takich czynników jak pomyłki podczas wprowadzania danych, usterki systemowe albo procesy scalania danych. Obecność duplikatów może obciążać modele i dawać niedokładne wyniki. Dlatego trzeba rozpoznać i wyeliminować powielone rekordy, żeby zapewnić dokładność i wiarygodność zbioru danych.

Istnieje wiele metod usuwania duplikatów w zbiorze danych. Najprostszą metodą jest porównanie wszystkich wierszy zbioru danych w celu zidentyfikowania powielonych rekordów. Jeśli dwa lub więcej wierszy ma te same wartości we wszystkich kolumnach, uważa się je za duplikaty. W niektórych przypadkach może być potrzebne porównanie tylko pewnego podzbioru kolumn, jeśli niektóre kolumny są bardziej podatne na duplikację.

Inną metodą wykrywania duplikatów jest użycie kolumny unikatowego identyfikatora. Jest to kolumna, która zawiera unikatowe wartości dla każdego rekordu, takie jak numer identyfikacyjny albo kombinacja unikatowych kolumn. Porównanie kolumny unikatowego identyfikatora pozwala wykryć i usunąć zduplikowane rekordy ze zbioru danych.

Po zidentyfikowaniu powielonych rekordów trzeba zdecydować, które rekordy zachować, a które usunąć. Jedną z opcji jest zachowanie pierwszego wystąpienia rekordu i usunięcie wszystkich kolejnych wystąpień. Innym podejściem jest zachowanie rekordów z najbardziej kompletnymi informacjami albo najnowszym znacznikiem czasowym.

Trzeba być świadomym, że usunięcie duplikatów może zmniejszyć rozmiar zbioru danych i wpłynąć na działanie modeli ML. Dlatego ważne jest rozważenie wpływu usunięcia duplikatów zarówno na zbiór danych, jak i model ML. W niektórych przypadkach konieczne może być zachowanie powielonych rekordów, jeśli zawierają one informacje, których nie można uzyskać z innych rekordów.

Standaryzacja i przekształcanie danych

Standaryzacja i przekształcanie danych to kluczowy etap w przygotowywaniu danych do zadań ML. Proces ten polega na skalowaniu i normalizacji liczbowych cech zbioru danych, aby ułatwić ich interpretację i porównywanie. Głównym celem standaryzacji i przekształcania danych jest zwiększenie dokładności i trafności modelu ML przez ograniczenie wpływu cech o zróżnicowanych skalach i zakresach. Popularna metoda standaryzacji danych jest nazywana standaryzacją Z lub normalizacją Z i polega na przekształceniu każdej cechy tak, aby miała średnią wartość oczekiwaną równą zeru i odchylenie standardowe równe jedności. Wzór na standaryzację Z jest następujący:

$$x' = (x - mean(x)) : std(x)$$

W tym równaniu x reprezentuje wartość cechy, $mean(x)$ oznacza średnią wartość cechy, $std(x)$ oznacza odchylenie standardowe cechy, a x' reprezentuje nową wartość przypisywaną cesze. Dzięki takiej standaryzacji danych zakres każdej cechy przesuwa się tak, że jego środek przypada na zero, co ułatwia porównywanie cech i zapobiega zdominowaniu analizy przez cechy o dużych wartościach.

Inną techniką przekształcania danych jest skalowanie min-max. Metoda ta przeskalowuje dane do spójnego zakresu wartości, zwykle od 0 do 1. Wzór na skalowanie min-max jest następujący:

$$x' = (x - min(x)) : (max(x) - min(x))$$

W tym równaniu x reprezentuje wartość cechy, $min(x)$ oznacza minimalną wartość cechy, a $max(x)$ oznacza maksymalną wartość cechy. Skalowanie min-max jest przydatne, kiedy precyzyjny rozkład danych nie ma większego znaczenia, ale dane trzeba ustandaryzować, żeby móc dokonywać znaczących porównań różnych cech.

Przekształcanie danych może również wiązać się ze zmienianiem ich rozkładu. Często stosowaną transformacją jest przekształcenie logarytmiczne, które ogranicza wpływ elementów odstających i skośności danych. Polega ono na obliczeniu logarytmu wartości cech, co pomaga znormalizować rozkład i ograniczyć wpływ wartości skrajnych.

Ogólnie rzecz biorąc, standaryzacja i przekształcanie danych są kluczowym etapem w procesie wstępnego przetwarzania danych na użytek ML. Poprzez skalowanie i normalizowanie cech możemy poprawić dokładność i trafność modeli ML, ułatwiając im interpretowanie i porównywanie danych.

Obsługa elementów odstających

Elementy odstające to punkty danych, które znacznie odbiegają od reszty obserwacji w zbiorze danych. Ich występowanie może wynikać z takich przyczyn jak błędy pomiarowe, uszkodzenie danych lub autentyczne skrajne wartości. Obecność elementów odstających

może mieć znaczny wpływ na wyniki modeli ML, ponieważ zniekształca dane i zakłóca związki między zmiennymi. Dlatego obsługa elementów odstających jest ważnym krokiem we wstępnym przetwarzaniu danych na użytek ML.

Istnieje kilka metod radzenia sobie z elementami odstającymi:

- **Usuwanie elementów odstających**. Proste podejście polega na wyeliminowaniu obserwacji zidentyfikowanych jako odstające. Trzeba jednak zachować przy tym ostrożność, ponieważ nadmierne usuwanie danych może doprowadzić do utraty cennych informacji i wypaczenia wyników analizy.
- **Przekształcenie danych**. Zastosowanie funkcji matematycznych, takich jak logarytm lub pierwiastek kwadratowy, może złagodzić wpływ elementów odstających. Na przykład obliczenie logarytmu zmiennej ogranicza wpływ skrajnych wartości, ponieważ skala logarytmiczna rośnie w mniejszym tempie niż pierwotne wartości.
- **Winsoryzacja**. Winsoryzacja to technika, która polega na zastępowaniu skrajnych wartości najbliższą najwyższą lub najniższą wartością w zbiorze danych. Zastosowanie tej metody pomaga w zachowaniu rozmiaru próby i ogólnego rozkładu danych.
- **Imputacja wartości**. Imputacja polega na zastępowaniu brakujących lub skrajnych wartości szacunkowymi wartościami obliczonymi na podstawie pozostałych obserwacji w zbiorze danych. Popularną techniką imputacji jest na przykład zastępowanie skrajnych wartości średnią lub medianą pozostałych obserwacji.
- **Używanie odpornych metod statystycznych**. Odporne metody statystyczne są mniej wrażliwe na elementy odstające, co prowadzi do dokładniejszych wyników nawet w obecności takich skrajnych wartości. Na przykład wybór mediany zamiast średniej może zmniejszyć wpływ elementów odstających na końcowe wyniki.

Trzeba podkreślić, że wybór metody obsługi elementów odstających powinien być dostosowany do unikatowej charakterystyki danych i do konkretnego problemu. Zasadniczo zaleca się stosować kombinację metod, aby kompleksowo rozwiązać problem elementów odstających, a kluczowe znaczenie ma ocena wpływu każdej metody na wyniki. Co więcej, ważne jest dokumentowanie każdej wykonanej czynności, ponieważ pozwala to na odtworzenie wyników i zapewnia wgląd w proces decyzyjny.

Poprawianie błędów

Poprawianie błędów na etapie przetwarzania wstępnego ma kluczowe znaczenie dla przygotowania danych na użytek ML. Błędy mogą pojawiać się z różnych przyczyn, takich jak pomyłki podczas wprowadzania danych, rozbieżności pomiarowe, niedokładność czujników albo usterki podczas transmisji. Poprawienie błędów pomaga zagwarantować, że model ML będzie trenowany na wiarygodnych i precyzyjnych danych, co zwiększy dokładność i niezawodność prognoz.

Istnieje kilka technik korygowania błędów w danych. Oto kilka najpopularniejszych:

- **Ręczna inspekcja**. Prostą metodą jest zbadanie zbioru danych i ręczne poprawienie błędów. Jest to często używana metoda, zwłaszcza w przypadku względnie małych zbiorów danych.
- **Metody statystyczne**. Metody statystyczne pozwalają efektywnie odkrywać i poprawiać błędy w danych. Na przykład kiedy dane odpowiadają znanemu rozkładowi, techniki statystyczne, takie jak standaryzacja Z, pozwalają wykryć elementy odstające, które można następnie usunąć lub zastąpić.
- **Metody ML**. Zastosowanie algorytmów ML ułatwia wykrywanie i poprawianie błędów w danych. Na przykład algorytmy klasteryzacji pomagają w identyfikowaniu punktów danych, które znacznie odbiegają od reszty zbioru danych. Zidentyfikowane w ten sposób punkty danych można poddać dalszej inspekcji i korekcji.
- **Wiedza dziedzinowa**. Wykorzystanie wiedzy dziedzinowej ma kluczowe znaczenie dla wykrywania błędów w danych. Na przykład podczas zbierania danych z czujników można identyfikować i poprawiać błędy przez uwzględnienie oczekiwanego zakresu wartości, które może generować czujnik.
- **Imputacja**. Imputacja jest metodą uzupełniania brakujących wartości w danych. Można to osiągnąć różnymi środkami, na przykład z użyciem metod statystycznych, takich jak imputacja średniej lub mediany, albo algorytmów ML, takich jak imputacja metodą k najbliższych sąsiadów.

Wybór techniki zależy od natury danych, rozmiaru zbioru danych oraz dostępnych zasobów.

Selekcja cech

Selekcja cech polega na wyborze najbardziej stosownych cech zbioru danych do zbudowania modelu ML. Celem jest zmniejszenie liczby cech bez znacznego pogorszenia dokładności modelu, co przekłada się na wyższą wydajność, szybszy trening i prostszą interpretację modelu.

Istnieje kilka podejść do selekcji cech. Przyjrzyjmy się im bliżej.

Metody filtrujące

Techniki te wykorzystują metody statystyczne do szeregowania cech według ich korelacji ze zmienną docelową. Do często używanych metod należą test chi kwadrat, informacja wzajemna i współczynniki korelacji. Cechy wybiera się następnie zgodnie ze wstępnie zdefiniowanym progiem.

Chi kwadrat

Test chi kwadrat to szeroko używana statystyczna metoda selekcji cech, która sprawdza się szczególnie dobrze w przypadku zmiennych kategorycznych. Test ten mierzy zależność między dwiema zmiennymi losowymi, dostarczając wartość p, która wskazuje prawdopodobieństwo uzyskania wyniku równie lub bardziej skrajnego niż rzeczywiste obserwacje.

W testowaniu hipotez test chi kwadrat pozwala ocenić, czy zgromadzone dane są zgodne z oczekiwanymi. Mały wynik testu chi kwadrat wskazuje dobre dopasowanie, a duży wynik — słabe dopasowanie. Wartość p mniejsza lub równa 0,05 prowadzi do odrzucenia hipotezy zerowej jako bardzo nieprawdopodobnej. I odwrotnie: wartość p wyższa niż 0,05 prowadzi do zaakceptowania albo przynajmniej nieodrzucenia hipotezy zerowej. Kiedy wartość jest zbliżona do 0,05, warto dokładniej zbadać hipotezę.

Jeśli chodzi o selekcję cech, test chi kwadrat ocenia związek między każdą cechą a zmienną docelową w zbiorze danych. Określa istotność cech na podstawie tego, czy istnieje statystycznie istotna różnica między zaobserwowaną a oczekiwaną częstością występowania cechy przy założeniu niezależności między cechą a zmienną docelową. Cechy z wysokim wskaźnikiem chi kwadrat wykazują większą zależność od zmiennej docelowej, co oznacza, że dostarczają więcej informacji w zadaniach klasyfikacji lub regresji. Wzór na wskaźnik chi kwadrat jest następujący:

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

W równaniu tym O_i reprezentuje wartość zaobserwowaną, a E_i — wartość oczekiwaną. Oblicza się różnicę między częstością zaobserwowaną a oczekiwaną, podnosi wynik do kwadratu, a następnie dzieli przez częstość oczekiwaną. Zsumowanie tych wartości dla wszystkich kategorii cechy daje ogólny wskaźnik chi kwadrat dla tej cechy.

Stopnie swobody testu zależą od liczby kategorii w cesze i liczby kategorii w zmiennej docelowej.

Przykładowym zastosowaniem selekcji cech opartej na teście chi kwadrat jest klasyfikacja tekstu, zwłaszcza w scenariuszu, w którym jako cechy wykorzystuje się obecność lub nieobecność określonych słów w dokumencie. Test chi kwadrat pomaga zidentyfikować słowa mocno związane ze szczególną klasą lub kategorią dokumentów, a następnie wykorzystać je jako cechy w modelu ML. W przypadku danych kategorycznych, zwłaszcza kiedy związek między cechami a zmienną docelową jest nieliniowy, test chi kwadrat jest skuteczną metodą selekcji cech. Jego użyteczność jednak maleje, kiedy cechy są ciągłe lub ściśle skorelowane; w takich przypadkach lepsze mogą być inne metody selekcji.

Informacja wzajemna

Informacja wzajemna to miara, która pozwala ocenić wzajemną zależność dwóch zmiennych losowych. W kontekście selekcji określa ona informacje na temat zmiennej docelowej, których dostarcza cecha. Podstawowa metodyka polega na obliczeniu informacji wzajemnej między każdą cechą a zmienną docelową i wybraniu tych cech, które mają najwyższe wskaźniki informacji wzajemnej.

Matematycznie informację wzajemną między dwiema dyskretnymi zmiennymi losowymi X i Y można zdefiniować następująco:

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

W równaniu tym $p(x, y)$ reprezentuje wspólną funkcję masy prawdopodobieństwa X i Y, a $p(x)$ i $p(y)$ oznaczają brzegowe funkcje masy prawdopodobieństwa X i Y.

W kontekście selekcji cech podczas obliczania informacji wzajemnej cechę traktuje się jako *X*, a zmienną docelową — jako *Y*. Po obliczeniu wskaźnika informacji wzajemnej dla każdej cechy można wybrać cechy z najwyższymi wskaźnikami.

Aby oszacować funkcje masy prawdopodobieństwa potrzebne do obliczenia informacji wzajemnej, można użyć metod histogramowych. Polega to na podziale zakresu każdej zmiennej na stałą liczbę przedziałów i oszacowaniu funkcji masy prawdopodobieństwa na podstawie częstości obserwacji w każdym przedziale. Można też użyć w tym celu jądrowego estymatora gęstości i obliczyć informację wzajemną na podstawie oszacowanych gęstości.

W praktycznych zastosowaniach informacji wzajemnej często używa się obok innych metod selekcji cech, takich jak metody oparte na teście chi kwadrat albo na korelacji, aby zwiększyć ogólną skuteczność procesu selekcji cech.

Współczynniki korelacji

Współczynniki korelacji wskazują siłę i kierunek liniowej relacji między dwiema zmiennymi. W kontekście selekcji cech przydają się do identyfikowania cech ściśle skorelowanych ze zmienną docelową, które w związku z tym są potencjalnie cennymi predyktorami.

Na potrzeby selekcji cech zwykle używa się współczynnika korelacji Pearsona, znanego też jako współczynnik *r*. Współczynnik *r* mierzy liniową zależność między dwiema zmiennymi ciągłymi i przyjmuje wartości od –1 (idealna korelacja ujemna) do 1 (idealna korelacja dodatnia), przy czym 0 wskazuje brak korelacji. Aby go obliczyć, należy podzielić kowariancję między dwiema zmiennymi przez iloczyn ich odchyleń standardowych, jak pokazuje poniższy wzór:

$$r = \frac{cov(X, Y)}{std(X) \cdot std(Y)}$$

W równaniu tym *X* i *Y* reprezentują dwie interesujące nas zmienne, *cov()* jest funkcją kowariancji, a *std()* oznacza funkcję odchylenia standardowego.

Wykorzystanie współczynnika *r* do selekcji cech polega na obliczeniu korelacji między każdą cechą a zmienną docelową, a następnie wybraniu cech o najwyższej wartości bezwzględnej współczynnika korelacji. Wysoka wartość bezwzględna współczynnika korelacji wskazuje ścisłą (czy to dodatnią, czy ujemną) korelację ze zmienną docelową. Interpretację wartości współczynnika korelacji Pearsona opisano w tabeli 3.1.

Tabela 3.1. Wartości współczynnika korelacji Pearsona i odpowiadające im stopnie korelacji

Wartość współczynnika korelacji Pearsona	Stopień korelacji
±1	Idealny
±0,50 – ±1	Wysoki
±0,30 – ±0,49	Umiarkowany
<0,29	Niski
0	Brak korelacji

Warto zauważyć, że współczynnik *r* nadaje się tylko do identyfikowania liniowych zależności między zmiennymi. Jeśli zależność jest nieliniowa albo jeśli jedna lub obie zmienne są kategoryczne, odpowiedniejsze mogą być inne współczynniki korelacji, takie jak współczynnik ρ Spearmana albo współczynnik τ Kendalla. Ponadto trzeba zachować ostrożność podczas interpretowania współczynników korelacji, ponieważ wysoka korelacja nie zawsze implikuje przyczynowość.

Metody opakowujące

Techniki te wybierają podzbiór cech przez iteracyjne trenowanie i testowanie modelu. Do powszechnie znanych metod należy selekcja w przód, eliminacja wsteczna oraz rekurencyjna eliminacja cech. Choć metody te są kosztowne obliczeniowe, mogą znacznie zwiększyć dokładność modelu.

Konkretnym przykładem metody opakowującej jest **rekurencyjna eliminacja cech** (ang. *recursive feature elimination*, **RFE**). RFE działa jak metoda eliminacji wstecznej, systematycznie usuwając najmniej ważne cechy, aż pozostanie wstępnie określona liczba cech. W każdej iteracji model ML jest trenowany na istniejących cechach i najmniej ważna cecha jest usuwana na podstawie jej wskaźnika ważności. Ten sekwencyjny proces powtarza się aż do osiągnięcia określonej liczby cech. Wskaźnik ważności cechy można obliczać różnymi metodami, na przykład na podstawie współczynnika korelacji w modelach liniowych albo poziomu cechy w drzewie decyzyjnym. RFE jest metodą kosztowną obliczeniowo, ale może być przydatna, jeśli liczba cech jest duża i trzeba ograniczyć przestrzeń cech. Alternatywnym podejściem jest wybieranie cech w procesie treningu, do czego wykorzystuje się metody zagnieżdżone.

Metody zagnieżdżone

Te metody wybierają cechy w procesie trenowania modelu. Do najpopularniejszych należą LASSO i regresja grzbietowa, drzewa decyzyjne i lasy losowe.

LASSO

LASSO (ang. *Least Absolute Shrinkage and Selection Operator*) to metoda regresji liniowej często używana do selekcji cech w uczeniu maszynowym. Wprowadza ona wyraz „kary" do standardowej funkcji straty. Kara ta zachęca model do zmniejszania współczynników mniej ważnych cech do zera, w rezultacie eliminując je z modelu. Metoda LASSO jest szczególnie cenna podczas pracy z danymi wielowymiarowymi, w których liczba cech znacznie przekracza liczbę próbek. W takich scenariuszach czasem trudno jest ustalić, które cechy najlepiej nadają się do przewidywania zmiennej docelowej. LASSO automatycznie identyfikuje najbardziej istotne cechy, jednocześnie zmniejszając współczynniki pozostałych.

Metoda LASSO znajduje rozwiązanie następującego problemu optymalizacji, który jest problemem minimalizacji:

$$\min_{w} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|w\|_1$$

W równaniu tym wektor *y* reprezentuje zmienną docelową, *X* oznacza macierz cech, *w* oznacza wektor współczynników regresji, λ jest hiperparametrem określającym

intensywność kary, a $||\mathbf{w}||_1$ oznacza normę ℓ_1 współczynników regresji (tzn. sumę ich wartości bezwzględnych).

Dodanie wyrazu kary ℓ_1 do funkcji celu skłania model do precyzyjnego wyzerowania niektórych współczynników, co zasadniczo eliminuje powiązane cechy z modelu. Siła kary zależy od hiperparametru λ, który można dostroić przez użycie walidacji krzyżowej.

W porównaniu z innymi metodami selekcji cech LASSO ma kilka zalet, m.in. zdolność do radzenia sobie ze skorelowanymi cechami oraz do jednoczesnego przeprowadzania selekcji cech i regresji. Ma jednak pewne ograniczenia, takie jak tendencja do wybierania tylko jednej cechy z grupy skorelowanych cech, a wydajność tej metody pogarsza się, jeśli liczba cech jest znacznie większa od liczby próbek.

Rozważmy zastosowanie metody LASSO do selekcji cech na potrzeby przewidywania cen domów. Wyobraź sobie zbiór danych zawierający szczegółowe informacje o domach — liczba sypialni, powierzchnia działki, rok budowy itd. — obok odpowiednich cen sprzedaży. Stosując LASSO, możemy zidentyfikować cechy, które mają największe znaczenie dla przewidywania ceny, a jednocześnie dopasować model regresji liniowej do zbioru danych. Wynikiem jest model gotowy do przewidywania ceny sprzedaży nowego domu na podstawie jego cech.

Regresja grzbietowa

Regresja grzbietowa, metoda regresji liniowej używana do selekcji cech, przypomina zwykłą regresję metodą najmniejszych kwadratów, ale dodaje wyraz kary do funkcji kosztu, aby zapobiec nadmiernemu dopasowaniu.

W regresji grzbietowej funkcję kosztu modyfikuje się przez dodanie wyrazu kary, który jest wprost proporcjonalny do kwadratu wielkości współczynników. Wyraz kary jest regulowany przez hiperparametr, często oznaczany literą λ lub α, który określa siłę regularyzacji. Kiedy α jest równe zeru, regresja grzbietowa redukuje się do zwykłej regresji metodą najmniejszych kwadratów.

Wpływ wyrazu kary przejawia się zmniejszaniem wielkości współczynników w kierunku zera. Ogranicza to ryzyko nadmiernego dopasowania, zniechęcając model do nadmiernego polegania na dowolnej pojedynczej cesze. W rezultacie wyraz kary pełni funkcję selektora cech, ograniczając znaczenie mniej istotnych funkcji.

Równanie funkcji straty w regresji grzbietowej jest następujące:

$$\min_{w} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \boldsymbol{\alpha}\|\boldsymbol{w}\|_2$$

Oto znaczenie poszczególnych wyrazów:

- N to liczba próbek w zbiorze treningowym.
- y to kolumnowy wektor wartości docelowych o rozmiarze N.
- X to macierz cech wejściowych.
- w to wektor estymowanych współczynników regresji.
- α to parametr regularyzacji, który określa siłę wyrazu kary. Jest to hiperparametr, który wymaga dostrojenia.

Pierwszy wyraz w funkcji straty mierzy błąd średniokwadratowy między wartościami przewidywanymi a rzeczywistymi. Drugim wyrazem jest kara ℓ_2, która zmniejsza współczynniki w kierunku zera. Algorytm regresji grzbietowej znajduje wartości współczynników regresji, które minimalizują tę funkcję straty. Poprzez dostrajanie parametru regularyzacji możemy sterować równowagą pomiędzy obciążeniem (biasem) a wariancją modelu, przy czym wyższe wartości α zwiększają regularyzację i ograniczają nadmierne dopasowanie.

Regresję grzbietową można wykorzystać do selekcji cech przez zbadanie wielkości współczynników wygenerowanych przez model. Cechy ze współczynnikami bliskimi zeru lub mniejszymi są mniej ważne i można usunąć je z modelu. Wartość α można dostroić z użyciem walidacji krzyżowej, aby znaleźć optymalną równowagę między złożonością a dokładnością modelu.

Jedną z głównych zalet regresji grzbietowej jest jej zdolność do radzenia sobie ze współliniowością, czyli silnymi korelacjami między niezależnymi zmiennymi. W takich przypadkach zwykła regresja metodą najmniejszych kwadratów może generować niestabilne estymacje współczynników, ale regresja grzbietowa pomaga ustabilizować estymacje i poprawić ogólną trafność modelu.

Wybór między LASSO a regresją grzbietową

Zarówno LASSO, jak i regresja grzbietowa to techniki regularyzacji, których używa się w regresji liniowej do zapobiegania nadmiernemu dopasowaniu przez nakładanie kar na współczynniki modelu. Choć obie metody mają zapobiegać nadmiernemu dopasowaniu, różnią się podejściem do nakładania kar na współczynniki.

Regresja grzbietowa dodaje do **sumy podniesionych do kwadratu błędów** (ang. *sum of squared errors*, **SSE**) wyraz kary, który jest proporcjonalny do kwadratu wielkości współczynników. Wyraz kary zależy od parametru regularyzacji (α), który określa, jak bardzo zmniejszane są współczynniki. Wyraz ten zmniejsza wartości współczynników w kierunku zera, ale nie ustawia ich dokładnie na zero. Dlatego regresja grzbietowa pozwala ograniczyć wpływ nieistotnych cech na model, ale nie eliminuje ich całkowicie.

Metoda LASSO również dodaje wyraz kary do SSE, ale jest on proporcjonalny do wartości bezwzględnej współczynników. Podobnie jak regresja grzbietowa, LASSO również ma parametr regularyzacji (λ), który określa, jak bardzo zmniejszane są współczynniki. LASSO ma jednak tę unikatową właściwość, że ustawia niektóre parametry dokładnie na zero, kiedy parametr regularyzacji jest wystarczająco wysoki. Dlatego metody tej można użyć do selekcji cech, ponieważ może ona wyeliminować nieistotne cechy przez ustawienie ich współczynników na zero.

Ogólnie rzecz biorąc, jeśli zbiór danych ma wiele cech i można przypuszczać, że tylko nieliczne okażą się ważne, regresja LASSO jest lepszym wyborem, ponieważ ustawi współczynniki nieistotnych cech na zero, prowadząc do prostszego i bardziej interpretowalnego modelu. Z drugiej strony, jeśli można przypuszczać, że większość cech w zbiorze danych będzie ważna, lepszym wyborem jest regresja grzbietowa, która zmniejszy mniej istotne cechy w kierunku zera, ale nie ustawi ich dokładnie na zero, zachowując wszystkie cechy w modelu.

Trzeba jednak podkreślić, że optymalny wybór między regresją grzbietową a LASSO zależy od konkretnego problemu i zbioru danych, więc często zaleca się, żeby wypróbować obie i porównać ich wyniki z użyciem technik walidacji krzyżowej.

Techniki ograniczania wymiarowości

Metody te przenoszą cechy w przestrzeń o mniejszej liczbie wymiarów przy zachowaniu jak największej ilości informacji. Do popularnych metod należy analiza głównych składowych (*principal component analysis*, PCA), liniowa analiza dyskryminacyjna (ang. *linear discriminant analysis*, **LDA**) oraz *t*-SNE.

PCA

PCA to technika szeroko stosowana w uczeniu maszynowym do ograniczania wymiarowości dużych zbiorów danych przy zachowaniu większości ważnych informacji. Podstawową koncepcją PCA jest przekształcenie zbioru skorelowanych zmiennych w zbiór nieskorelowanych zmiennych znanych jako składowe główne.

Celem PCA jest zidentyfikowanie kierunków maksymalnej wariancji w danych i rzutowanie danych na te kierunki w celu zmniejszenia wymiarowości danych. Składowe główne sortuje się według ilości wariancji, którą wyjaśniają, przy czym pierwsza składowa główna wyjaśnia najwięcej wariancji w danych.

Algorytm PCA składa się z następujących kroków:

1. **Standaryzacja danych**. PCA wymaga, aby dane były ustandaryzowane, tzn. każda cecha musi mieć zerową średnią i jednostkową wariancję.
2. **Obliczenie macierzy kowariancji**. Macierz kowariancji to kwadratowa macierz, która opisuje liniowe relacje między parami cech w danych.
3. **Obliczenie wektorów i wartości własnych macierzy kowariancji**. Wektory własne reprezentują podstawowe kierunki najwyższej wariancji w zbiorze danych, a wartości własne określają stopień wariancji wyjaśnianej przez każdy wektor własny.
4. **Wybranie liczby składowych głównych**. Liczbę składowych głównych do zachowania można ustalić przez przeanalizowanie wartości własnych i wybranie *k* wektorów, które wyjaśniają najwięcej wariancji.
5. **Rzutowanie danych na wybrane składowe główne**. Pierwotne dane rzutuje się na wybrane składowe główne, uzyskując reprezentację danych o niższej liczbie wymiarów.

PCA można użyć do selekcji cech poprzez wybranie *k* składowych głównych, które wyjaśniają najwięcej wariancji w danych. Może to przydać się do ograniczenia wymiarowości wysokowymiarowych zbiorów danych i poprawienia wydajności modeli uczenia maszynowego. Trzeba jednak zauważyć, że PCA nie zawsze prowadzi do wyższej wydajności, zwłaszcza jeśli dane są już niskowymiarowe albo jeśli cechy nie są ściśle skorelowane. Trzeba też rozważyć interpretowalność składowych głównych, ponieważ nie zawsze odpowiadają one znaczącym cechom w danych.

LDA

LDA to technika ograniczania wymiarowości używana do selekcji cech w uczeniu maszynowym. Często wykorzystuje się ją w zadaniach klasyfikacji do ograniczenia liczby cech przez przeniesienie ich do niskowymiarowej przestrzeni z zachowaniem jak największej ilości informacji odróżniających klasy.

W LDA celem jest znalezienie liniowej kombinacji pierwotnych cech, która maksymalizuje separację między klasami. Wejściem LDA jest zbiór danych z opatrzonymi etykietami wektorami, przy czym każdy przykład jest wektorem cech z odpowiednią etykietą klasy. Wyjściem LDA jest zbiór liniowych kombinacji pierwotnych cech, których można użyć jako nowych cech w modelu uczenia maszynowego.

Aby przeprowadzić LDA, najpierw należy obliczyć wektor średni i macierz kowariancji każdej klasy. Następnie oblicza się ogólny wektor średni i macierz kowariancji na podstawie wektorów średnich i macierzy kowariancji poszczególnych klas. Celem jest rzutowanie danych na przestrzeń o niższej liczbie wymiarów przy zachowaniu informacji o klasach. Osiąga się to przez obliczenie wektorów i wartości własnych macierzy kowariancji, posortowanie ich w malejącej kolejności wartości własnych i wybranie *k* wektorów własnych, które odpowiadają *k* najwyższym wartościom własnym. Wybrane wektory własne stanowią bazę nowej przestrzeni cech.

Algorytm LDA można podsumować następująco:

1. Oblicz wektor średni każdej klasy.
2. Oblicz macierz kowariancji każdej klasy.
3. Oblicz ogólny wektor średni i ogólną macierz kowariancji.
4. Oblicz macierz rozrzutu między klasami.
5. Oblicz macierz rozrzutu w obrębie klasy.
6. Oblicz wektory i wartości własne macierzy z następującego wzoru:

$$\mathbf{S}_w^{-1} \cdot \mathbf{S}_b$$

   $\mathbf{S}_w$ jest tu macierzą rozrzutu w obrębie klasy, a $\mathbf{S}_b$ — macierzą rozrzutu między klasami.
7. Wybierz *k* wektorów własnych z najwyższymi wartościami własnymi jako nową przestrzeń cech.

Metoda LDA jest szczególnie użyteczna, kiedy liczba cech jest duża, a liczba przykładów mała. Można używać jej do wielu różnych celów, takich jak rozpoznawanie obrazów, rozpoznawanie mowy i NLP. Zakłada ona jednak, że klasy mają rozkład normalny, a macierze kowariancji klas są równe, co w praktyce nie zawsze jest spełnione.

t-SNE

t-SNE to technika ograniczania wymiarowości służąca do wizualizowania wysokowymiarowych danych w przestrzeni o niższej liczbie wymiarów, często używana do selekcji cech. Została opracowana przez Laurensa van der Maatena i Geoffreya Hintona w 2008 r.

Podstawową koncepcją *t*-SNE jest zachowanie podobieństw między parami punktów w przestrzeni o niższej liczbie wymiarów zamiast zachowywania odległości między nimi. Innymi słowy, metoda ta próbuje zachować lokalną strukturę danych, jednocześnie odrzucając strukturę globalną. Może to być przydatne w sytuacjach, w których trudno jest zwizualizować wysokowymiarowe dane, ale między punktami danych mogą istnieć znaczące relacje i wzorce.

t-SNE zaczyna od obliczenia podobieństwa między każdą parą punktów w wysokowymiarowej przestrzeni. Podobieństwo zwykle mierzy się za pomocą jądra gaussowskiego, które nadaje wyższe wagi pobliskim punktom, a niższe odległym. Macierz podobieństw przekształca się następnie w rozkład prawdopodobieństwa z użyciem funkcji softmax. Rozkład ten wykorzystuje się do utworzenia przestrzeni niskowymiarowej, zwykle 2D lub 3D.

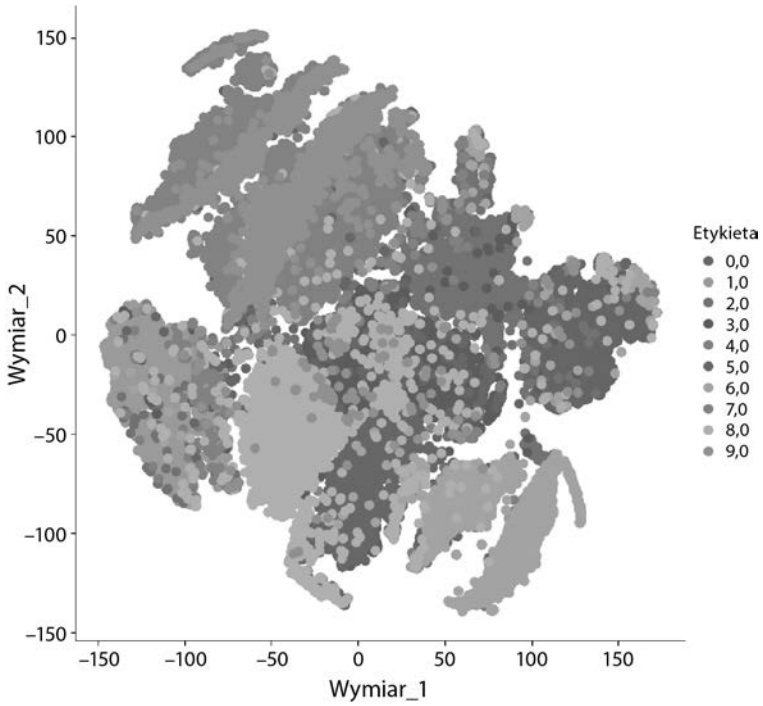
W przestrzeni niskowymiarowej *t*-SNE ponownie oblicza podobieństwa między każdą parą punktów danych, ale tym razem z wykorzystaniem rozkładu t Studenta zamiast rozkładu Gaussa. Rozkład t ma cięższe ogony niż rozkład Gaussa, co pomaga zachować lokalną strukturę danych. *t*-SNE następnie modyfikuje położenie punktów w niskowymiarowej przestrzeni tak, aby zminimalizować różnice między podobieństwami par w wysokowymiarowej przestrzeni i podobieństwami par w niskowymiarowej przestrzeni.

t-SNE to bardzo użyteczna technika do wizualizowania wysokowymiarowych danych poprzez zredukowanie ich do przestrzeni niskowymiarowej. Zwykle nie używa się jej jednak do selekcji cech, ponieważ jej podstawowym przeznaczeniem jest tworzenie wizualizacji złożonych zbiorów danych.

Zamiast tego *t*-SNE można wykorzystać do zidentyfikowania klastrów punktów danych, które mają podobne cechy, co może przydać się do identyfikowania grup cech, które są ważne dla konkretnego zadania. Przypuśćmy, że masz zbiór danych z informacjami demograficznymi o klientach oraz ich historiami zakupów i chcesz zidentyfikować grupy klientów, które są podobne pod względem zachowań zakupowych. Mógłbyś użyć *t*-SNE, aby zredukować wysokowymiarową przestrzeń cech do dwóch wymiarów, a następnie przedstawić uzyskane dane na wykresie punktowym. Badając ten wykres, mógłbyś zidentyfikować klastry klientów o podobnych zachowaniach zakupowych i wykorzystać te informacje w procesie selekcji cech. Przykład użycia techniki *t*-SNE na zbiorze danych MNIST pokazano na rysunku 3.1.

Warto zaznaczyć, że *t*-SNE to przede wszystkim narzędzie wizualizacyjne, którego nie należy używać jako jedynej metody selekcji cech. Można natomiast użyć go w połączeniu z innymi technikami, takimi jak LDA lub PCA, aby lepiej zrozumieć podstawową strukturę danych.

Wybór metody selekcji cech zależy od natury danych, rozmiaru zbioru danych, złożoności modelu i dostępnych zasobów obliczeniowych. Po wyborze cech należy uważnie ocenić działanie modelu, aby upewnić się, że nie doszło do utraty ważnych informacji. Innym ważnym procesem jest inżynieria cech, która polega na przekształcaniu lub wybieraniu cech na użytek modeli uczenia maszynowego.



Rysunek 3.1. Użycie techniki t-SNE na zbiorze danych MNIST

Inżynieria cech

Inżynieria cech to proces wybierania i przekształcania cech oraz wyodrębniania ich z surowych danych w celu poprawienia działania modeli uczenia maszynowego. Cechy to indywidualne mierzalne własności albo charakterystyki danych, których można używać do prognozowania lub klasyfikacji.

Jedną z technik inżynierii cech jest selekcja cech, która polega na wybieraniu podzbioru istotnych cech z pierwotnego zbioru danych w celu poprawienia dokładności modelu i ograniczenia jego złożoności. Można robić to metodami statystycznymi, takimi jak analiza korelacji albo szeregowanie ważności cech z użyciem drzew decyzyjnych lub lasów losowych.

Inną techniką inżynierii cech jest wyodrębnianie cech, które polega na przekształcaniu surowych danych w nowy zbiór cech, które mogą być bardziej użyteczne dla modelu. Podstawową różnicą między selekcją cech a inżynierią cech jest podejście: selekcja cech zachowuje podzbiór pierwotnych cech bez modyfikowania wybranych cech, podczas gdy algorytmy inżynierii cech rekonfigurują dane i przekształcają je w nową przestrzeń cech. Do inżynierii cech można wykorzystać takie techniki jak ograniczanie wymiarowości, PCA lub *t*-SNE. Selekcja i inżynieria cech zostały wyjaśnione szczegółowo w poprzednim punkcie.

Kolejną ważną techniką w inżynierii cech jest skalowanie cech, które polega na sprowadzeniu wartości cech do jednakowego zakresu, zwykle od 0 do 1 albo od –1 do 1. Robi się to, aby jedne cechy nie zdominowały drugich, a algorytm mógł szybko osiągnąć konwergencję podczas treningu. Kiedy cechy w zbiorze danych mają różne skale, może to prowadzić do problemów z niektórymi algorytmami uczenia maszynowego, które są wrażliwe na względne wielkości cech. Skalowanie cech pomaga rozwiązać ten problem poprzez konwersję wszystkich cech do podobnej skali. Do często używanych metod skalowania cech należy skalowanie min-max, skalowanie oparte na standaryzacji Z oraz skalowanie według maksymalnej wartości bezwzględnej.

Metody te krótko opisano poniżej:

- **Skalowanie min-max**. Technika ta, zwana również normalizacją, skaluje wartości cech tak, aby mieściły się w zadanym zakresie, zwykle od 0 do 1 (w przypadku zwykłych modeli uczenia maszynowego, czasem od –1 do 1 w przypadku modeli uczenia głębokiego). Wzór na skalowanie min-max jest następujący:

$$x_{przeskalowane} = (x - min(x)) : (max(x) - min(x))$$

  W tym równaniu x jest pierwotną wartością cechy, $min(x)$ oznacza minimalną wartość cechy, a $max(x)$ oznacza maksymalną wartość cechy.

- **Standaryzacja**. Technika ta przekształca wartości cech tak, aby miały średnią równą zeru i odchylenie standardowe równe jedności. Standaryzacja jest mniej wrażliwa na wartości odstające niż skalowanie min-max. Wzór na standaryzację jest następujący:

$$x_{przeskalowane} = (x - mean(x)) : std(x)$$

  W tym równaniu x reprezentuje wartość cechy, $mean(x)$ oznacza średnią wartość cechy, a $std(x)$ oznacza odchylenie standardowe cechy.

- **Skalowanie odporne na wartości odstające**. Ta technika jest podobna do standaryzacji, ale używa mediany i **rozstępu ćwiartkowego** (ang. *interquartile range*, **IQR**) zamiast średniej i odchylenia standardowego. Wzór na skalowanie odporne na wartości odstające jest następujący:

$$x_{przeskalowane} = (x - median(x)) : (Q3(x) - Q1(x))$$

  W tym równaniu x jest pierwotną wartością cechy, $median(x)$ jest medianą cechy, $Q1(x)$ jest pierwszym kwartylem cechy, a $Q3(x)$ jest trzecim kwartylem cechy.

- **Przekształcenie logarytmiczne**. Techniki tej używa się, gdy dane cechują się dużą skośnością albo mają długi ogon. Poprzez obliczenie logarytmu wartości cech można zmienić rozkład w bardziej normalny lub symetryczny, co może poprawić działanie niektórych algorytmów uczenia maszynowego. Wzór na przekształcenie logarytmiczne pokazano poniżej:

$$x_{przekształcone} = log(x)$$

  W równaniu tym x jest pierwotną wartością cechy.

- **Przekształcenie potęgowe**. Technika ta jest podobna do przekształcenia logarytmicznego, ale umożliwia szerszy zakres transformacji.

Najpopularniejszym przekształceniem potęgowym jest przekształcenie Boxa-Coxa, które podnosi wartości cech do potęgi określonej z użyciem estymacji maksymalnej wiarygodności. Wzór na przekształcenie Boxa-Coxa pokazano poniżej:

$$x_{przekształcone} = \frac{x^{\lambda} - 1}{\lambda}$$

W równaniu tym *x* jest pierwotną wartością cechy, a λ jest parametrem potęgowym oszacowanym metodą maksymalnej wiarygodności.

Są to najpopularniejsze metody skalowania cech w uczeniu maszynowym. Wybór metody zależy od rozkładu danych, używanego algorytmu uczenia maszynowego oraz konkretnych wymagań problemu.

Ostatnią techniką inżynierii cech jest konstruowanie cech, które polega na tworzeniu nowych cech poprzez łączenie lub przekształcanie istniejących. Można wykorzystać do tego takie techniki jak rozwinięcie wielomianowe, przekształcenie logarytmiczne albo wyrazy interakcji.

Rozwinięcie wielomianowe

Rozwinięcie wielomianowe to technika konstrukcji cech, która polega na tworzeniu nowych cech jako wielomianowych kombinacji istniejących cech. Techniki tej używa się w uczeniu maszynowym do modelowania nieliniowych związków między cechami a zmienną docelową.

W rozwinięciu wielomianowym tworzy się nowe cechy przez podnoszenie istniejących cech do różnych potęg oraz mnożenie ich przez siebie. Przypuśćmy na przykład, że mamy jedną cechę *x*. Możemy utworzyć nową cechę, podnosząc *x* do kwadratu (x^2). Możemy też utworzyć cechy wielomianowe wyższego rzędu, podnosząc *x* do jeszcze wyższych potęg, takich jak x^3, x^4 itd. Ogólnie rzecz biorąc, możemy utworzyć cechy wielomianowe stopnia *d*, obliczając wszystkie możliwe kombinacje iloczynów i potęg pierwotnych cech aż do stopnia *d*.

Oprócz tworzenia cech wielomianowych z jednej cechy możemy też utworzyć je z wielu cech. Przypuśćmy na przykład, że mamy dwie cechy, x_1 i x_2. Aby utworzyć nowe cechy, możemy pomnożyć je przez siebie (x_1x_2) i podnieść je do różnych potęg (x_1^2, x_2^2 itd.). Ponownie: możemy utworzyć cechy wielomianowe dowolnego stopnia jako wszystkie możliwe kombinacje iloczynów i potęg pierwotnych cech.

Kiedy używamy rozwinięcia wielomianowego, musimy pamiętać, że może ono szybko prowadzić do dużej liczby cech, zwłaszcza w przypadku wielomianów wyższego rzędu. Może to sprawić, że wynikowy model będzie bardziej złożony i trudniejszy w interpretacji, a także prowadzić do nadmiernego dopasowania, jeśli nie zachowamy kontroli nad liczbą cech. Aby rozwiązać ten problem, często używa się technik regularyzacji albo metod selekcji cech, tak by wybrać podzbiór cech wielomianowych, które niosą najwięcej informacji.

Ogólnie rzecz biorąc, rozwinięcie wielomianowe to użyteczna technika konstrukcji cech, która pomaga uchwycić złożone, nieliniowe relacje między cechami a zmienną

docelową. Należy jednak używać jej ostrożnie i w połączeniu z odpowiednią regularyzacją lub selekcją cech, aby uniknąć nadmiernego dopasowania i zachować interpretowalność modelu.

Przypuśćmy, że w problemie regresji mamy zbiór danych z pojedynczą cechą x i chcemy dopasować do niego model, który wychwyci relację między x a zmienną docelową y. Jednak relacja między x a y może nie być liniowa, a wówczas prosty liniowy model może okazać się niewystarczający. W takim przypadku rozwinięcie wielomianowe pozwoli utworzyć dodatkowe cechy odzwierciedlające nieliniowy związek między x a y.

Zilustrujmy to na przykładzie. Powiedzmy, że mamy zbiór danych z pojedynczą cechą x oraz zmienną docelową y i chcemy dopasować model regresji wielomianowej. Celem jest znalezienie funkcji $f(x)$, która minimalizuje różnicę między przewidywaną a rzeczywistą wartością y.

Za pomocą rozwinięcia wielomianowego możemy utworzyć dodatkowe cechy oparte na x, takie jak x^2, x^3 itd. Można to zrobić z użyciem bibliotek, takich jak `scikit–learn`, która zawiera funkcję `PolynomialFeatures` automatycznie generującą cechy wielomianowe określonego stopnia.

Kiedy dodamy te cechy wielomianowe, model stanie się bardziej ekspresywny i będzie mógł uchwycić nieliniowy związek między x a y. Trzeba jednak uważać, żeby nadmiernie nie dopasować modelu do danych, ponieważ dodanie zbyt wielu cech wielomianowych może prowadzić do modelu, który jest nadmiernie złożony i źle radzi sobie z nowymi, niewidzianymi wcześniej danymi.

Przekształcenie logarytmiczne

Przekształcenie logarytmiczne to popularna technika inżynierii cech, której używa się do wstępnego przetwarzania danych. Celem przekształcenia logarytmicznego jest zmniejszenie skośności danych przez przeliczenie cech funkcją logarytmiczną. Technika ta jest szczególnie przydatna w przypadku cech, które są niezrównoważone, na przykład mają długi ogon wysokich wartości.

Przekształcenie logarytmiczne jest zdefiniowane jako równanie, które oblicza logarytm naturalny danych:

$$y = log(x)$$

W równaniu tym y reprezentuje przekształcone dane, a x — pierwotne dane. Funkcja logarytmiczna odwzorowuje pierwotne dane na nową przestrzeń, w której relacja między wartościami pozostaje zachowana, ale skala jest skompresowana. Przekształcenie logarytmiczne jest szczególnie użyteczne w przypadku cech o dużych zakresach albo takich, które mają rozkład wykładniczy, takich jak ceny produktów albo zarobki.

Jedną z zalet przekształcenia logarytmicznego jest to, że pomaga ono znormalizować dane i dostosować je do wymagań niektórych algorytmów uczenia maszynowego, które zakładają rozkład normalny danych. Ponadto przekształcenie logarytmiczne może ograniczyć wpływ wartości odstających, a przez to poprawić działanie niektórych modeli.

Trzeba zauważyć, że przekształcenie logarytmiczne nie jest odpowiednie dla wszystkich typów danych. Jeśli na przykład dane zawierają wartości zerowe lub ujemne,

przekształcenia logarytmicznego nie można zastosować bezpośrednio. W takich przypadkach można zastosować zmodyfikowane przekształcenie logarytmiczne, na przykład przez dodanie stałej przed obliczeniem logarytmu. Ogólnie rzecz biorąc, przekształcenie logarytmiczne to przydatna technika inżynierii cech, która pomaga poprawić działanie modeli uczenia maszynowego, zwłaszcza w przypadku pracy ze skośnymi albo rozłożonymi wykładniczo danymi.

Podsumowując, inżynieria cech jest kluczowym etapem w procesie uczenia maszynowego, ponieważ ma znaczny wpływ na trafność i interpretowalność wynikowych modeli. Efektywna inżynieria cech wymaga wiedzy domenowej, kreatywności oraz iteracyjnego procesu testowania i dopracowywania różnych technik aż do zidentyfikowania optymalnego zbioru cech.

Wyrazy interakcji

W konstruowaniu cech termin „wyrazy interakcji" odnosi się to tworzenia nowych cech przez łączenie dwóch lub wielu istniejących cech poprzez mnożenie, dzielenie lub inne operacje matematyczne. Te nowe funkcje odzwierciedlają interakcje lub relacje między pierwotnymi cechami i mogą zwiększać dokładność modeli uczenia maszynowego.

Na przykład w zbiorze cech z cenami nieruchomości mogą znajdować się takie cechy jak liczba sypialni, liczba łazienek oraz powierzchnia w metrach kwadratowych. Same w sobie cechy te dostarczają pewnych informacji o cenie nieruchomości, ale nie odzwierciedlają skutków interakcji między cechami. Jeśli jednak utworzymy wyraz interakcji między liczbą sypialni a powierzchnią, będziemy mogli uchwycić ideę, że większe nieruchomości z większą liczbą łazienek zwykle są droższe niż mniejsze nieruchomości z tą samą liczbą łazienek.

W praktyce wyrazy interakcji tworzy się przez mnożenie lub dzielenie dwóch cech. Jeśli na przykład mamy dwie cechy, *x* i *y*, możemy utworzyć wyraz interakcji przez ich pomnożenie: *xy*. Możemy też utworzyć wyraz interakcji przez podzielenie jednej cechy przez drugą: *x*:*y*.

Podczas tworzenia wyrazów interakcji trzeba zastanowić się, które cechy warto połączyć i w jaki sposób. Oto kilka typowych technik:

- **Wiedza domenowa**. Użyj wiedzy domenowej lub intuicji eksperta, aby zidentyfikować cechy, które prawdopodobnie wchodzą ze sobą w interakcję.
- **Kombinacje par**. Utwórz wyrazy interakcji przez połączenie wszystkich par cech w zbiorze danych. Jest to kosztowne obliczeniowo, ale może pomóc w zidentyfikowaniu potencjalnych efektów interakcji.
- **PCA**. Użyj PCA, aby zidentyfikować najważniejsze kombinacje cech, i utwórz wyrazy interakcji na podstawie tych kombinacji.

Ogólnie rzecz biorąc, wyrazy interakcji to przydatne narzędzie do konstruowania cech, które pomaga uchwycić złożone relacje między cechami i poprawić działanie modeli uczenia maszynowego. Trzeba jednak używać ich ostrożnie, ponieważ zbyt duża liczba albo zły dobór wyrazów może prowadzić do nadmiernego dopasowania albo mniejszej interpretowalności modelu.

Typowe modele uczenia maszynowego

W tym podrozdziale omówimy niektóre najpopularniejsze modele uczenia maszynowego, a także ich zalety i wady. Informacje te pomogą Ci wybrać model najlepiej pasujący do problemu, a także ulepszyć zaimplementowany model.

Regresja liniowa

Regresja liniowa to algorytm uczenia nadzorowanego, który służy do modelowania związku między zmienną zależną a jedną lub wieloma zmiennymi niezależnymi. Zakłada on liniowy związek między cechami wejściowymi a wartością wyjściową. Celem regresji liniowej jest znalezienie linii najlepszego dopasowania, która przewiduje wartość zmiennej zależnej na podstawie zmiennych niezależnych.

Równanie prostej regresji liniowej z jedną zmienną niezależną (nazywane też **prostym równaniem liniowym**) jest następujące:

$$y = mx+b$$

Oto znaczenie poszczególnych wyrazów:

- y to zmienna zależna (zmienna, którą chcemy przewidywać),
- x to zmienna niezależna (zmienna wejściowa),
- m to współczynnik kierunkowy (nachylenie) linii (który określa, jak bardzo zmienia się y, kiedy zmienia się x),
- b to wyraz wolny (punkt, w którym linia przecina oś Y, kiedy $x = 0$).

Celem regresji liniowej jest znalezienie wartości m i b, które minimalizują różnicę między przewidywanymi a rzeczywistymi wartościami zmiennej zależnej. Różnicę tę zwykle mierzy się z użyciem funkcji kosztu, takiej jak błąd średniokwadratowy albo średni błąd bezwzględny.

Wielokrotna regresja liniowa jest rozszerzeniem prostej regresji liniowej i wykorzystuje wiele zmiennych niezależnych. Równanie wielokrotnej regresji liniowej pokazano poniżej:

$$y = b_0+b_1x_1+b_2x_2+...+b_nx_n$$

Oto znaczenie poszczególnych wyrazów:

- y to zmienna zależna,
- x_1, x_2, … x_n to zmienne niezależne,
- b_0 to wyraz wolny (punkt, w którym wszystkie zmienne niezależne są równe 0),
- b_1, b_2, … b_n to współczynniki kierunkowe (które określają, jak bardzo zmienia się y, kiedy zmienia się każda zmienna niezależna).

Podobnie jak w przypadku prostej regresji liniowej, celem wielokrotnej regresji liniowej jest znalezienie wartości b_0, b_1, b_2, …, b_n, które minimalizują różnicę między przewidywanymi a rzeczywistymi wartościami zmiennej zależnej.

Zalety regresji liniowej są następujące:

- Jest prosta i zrozumiała.
- Można używać jej do modelowania szerokiej gamy relacji między zmiennymi zależnymi i niezależnymi.
- Jest efektywna obliczeniowo, a przez to szybka i odpowiednia do przetwarzania dużych zbiorów danych.
- Dostarcza interpretowalnych wyników, umożliwiając analizowanie wpływu każdej zmiennej niezależnej na zmienną zależną.

Wady regresji liniowej są następujące:

- Zakłada związek liniowy między cechami wejściowymi a wartością wyjściową, który nie zawsze istnieje w rzeczywistych danych.
- Nie wychwytuje złożonych, nieliniowych relacji między cechami wejściowymi a wartością wyjściową.
- Jest wrażliwa na wartości odstające i wpływowe obserwacje, które mogą pogarszać dokładność modelu.
- Zakłada, że błędy mają rozkład normalny i stałą wariancję, co w praktyce nie zawsze jest prawdą.

Regresja logistyczna

Regresja logistyczna to popularny algorytm uczenia maszynowego używany w problemach klasyfikacji. W przeciwieństwie do regresji liniowej, która służy do przewidywania wartości ciągłych, regresji logistycznej używa się do przewidywania wyników dyskretnych, zwykle binarnych (0 lub 1).

Celem regresji logistycznej jest oszacowanie prawdopodobieństwa pewnego wyniku na podstawie jednej lub wielu zmiennych wejściowych. Wyjściem regresji logistycznej jest wskaźnik prawdopodobieństwa, który można przekształcić w binarną etykietę klasy przez zastosowanie wartości progowej. Wartość progową można wyregulować, aby zapewnić równowagę między precyzją a przywołaniem na podstawie konkretnych wymagań problemu.

Model regresji logistycznej zakłada, że związek między zmiennymi wejściowymi a zmienną wyjściową jest liniowy w przestrzeni logitowej (logarytmu szansy). Funkcja logitowa jest zdefiniowana następująco:

$$logit(p) = log(p:(1-p))$$

W równaniu tym p jest prawdopodobieństwem wyniku pozytywnego (tzn. prawdopodobieństwem, że dojdzie do zdarzenia).

Model regresji logistycznej można sformułować matematycznie w następujący sposób:

$$logit(p) = \beta_0+\beta_1x_1+ \beta_2x_2+ \ldots + \beta_nx_n$$

W równaniu tym β_0, β_1, β_2, … β_n są współczynnikami modelu, x_1, x_2, … x_n to zmienne wejściowe, a $logit(p)$ to logitowa funkcja prawdopodobieństwa pozytywnego wyniku.

Model regresji logistycznej trenuje się na zbiorze danych z przykładami opatrzonymi etykietami. Każdy przykład składa się ze zbioru zmiennych wejściowych oraz binarnej etykiety, która wskazuje, czy nastąpił wynik pozytywny, czy nie. Współczynniki modelu oblicza się przez estymację maksymalnej wiarygodności, która znajduje wartości współczynników maksymalizujące prawdopodobieństwo zaobserwowania danych.

Zalety regresji logistycznej są następujące:

- **Interpretowalność**. Współczynniki modelu można interpretować jako zmianę logarytmu szans na pozytywny wynik przy jednostkowej zmianie odpowiedniej zmiennej wejściowej, co ułatwia zrozumienie wpływu każdej zmiennej wejściowej na przewidywane prawdopodobieństwo pozytywnego wyniku.
- **Efektywność obliczeniowa**. Regresja logistyczna to prosty algorytm, który można szybko wytrenować na dużych zbiorach danych.
- **Działa dobrze na małych zbiorach danych**. Regresja logistyczna może być skuteczna nawet wtedy, gdy liczba obserwacji jest mała, pod warunkiem że zmienne wejściowe są istotne dla zadania.

Wady regresji logistycznej są następujące:

- **Zakłada liniowość**. Regresja logistyczna zakłada liniowy związek między zmiennymi wejściowymi a logitem prawdopodobieństwa pozytywnego wyniku, a warunek ten nie zawsze jest spełniony w rzeczywistych zbiorach danych.
- **Jest podatna na nadmierne dopasowanie**. Jeśli liczba zmiennych wejściowych jest duża w porównaniu z liczbą obserwacji, model może nadmiernie dopasować się do danych treningowych i słabo generalizować na nowe dane.
- **Nie nadaje się do problemów nieliniowych**. Regresja logistyczna jest algorytmem liniowym, więc nie nadaje się do problemów, w których związek między zmiennymi wejściowymi a zmienną wyjściową jest nieliniowy.

Drzewa decyzyjne

Drzewa decyzyjne to algorytm uczenia nadzorowanego używany do klasyfikacji i analizy regresyjnej. Drzewo decyzyjne składa się z szeregu węzłów, które reprezentują punkty decyzyjne. Każdy ma jedną lub wiele gałęzi, które prowadzą albo do innego punktu decyzyjnego, albo do ostatecznej prognozy.

W problemie klasyfikacji każdy liść drzewa reprezentuje etykietę klasy, a w problemie regresji — wartość liczbową. Proces budowania drzewa decyzyjnego polega na wybieraniu sekwencji atrybutów, które najlepiej dzielą dane na podzbiory bardziej homogeniczne względem zmiennej docelowej. Proces ten zwykle powtarza się rekurencyjnie dla każdego podzbioru aż do osiągnięcia kryterium zatrzymania, takiego jak minimalna liczba instancji w każdym podzbiorze albo maksymalna głębokość drzewa.

Równania drzew decyzyjnych obliczają zysk informacyjny (albo inne kryterium podziału, takie jak wskaźnik Giniego albo entropia) dla każdego potencjalnego podziału w każdym

węźle decyzyjnym. Atrybut o największym zysku informacyjnym jest wybierany jako kryterium podziału w tym węźle. Koncepcyjną formułę zysku informacyjnego pokazano poniżej:

Zysk informacyjny = *entropia*(*rodzic*) – [*ważona średnia entropii dzieci rodzica*]

Entropia oznacza tu miarę „nieczystości" albo losowości systemu. W kontekście drzew decyzyjnych entropii używa się do mierzenia nieczystości węzła w drzewie.

Entropię węzła oblicza się w następujący sposób:

$$Entropia = \sum_{i=1}^{c} -p_i \log_2 p_i$$

W równaniu tym c jest liczbą klas, a p_i jest proporcją próbek, które należą do klasy i w węźle.

Entropia węzła mieści się w zakresie od 0 do 1, przy czym 0 wskazuje czysty węzeł (tzn. wszystkie próbki należą do tej samej klasy), a 1 wskazuje węzeł równomiernie rozdzielony między wszystkie klasy.

W drzewie decyzyjnym entropii węzła używa się do określenia kryterium podziału drzewa. Chodzi o to, aby podzielić węzeł na dwa lub więcej węzłów potomnych, tak aby entropia węzłów potomnych była niższa od entropii rodzica. Jako najlepszy podział wybiera się ten, który daje najniższą entropię.

Zauważ, że wybór następnego węzła w drzewie decyzyjnym zależy od używanego algorytmu, np. CART, ID3 lub C4.5. Tutaj wyjaśniono algorytm CART, który dzieli dane na podstawie miary nieczystości Giniego i entropii.

Zaletą używania entropii jako kryterium podziału jest to, że sprawdza się ona zarówno w binarnych, jak i wieloklasowych problemach klasyfikacji. Jest też względnie efektywna obliczeniowo w porównaniu z innymi kryteriami podziału. Jednak wadą entropii jest to, że ma tendencję do tworzenia stronniczych drzew, ukierunkowanych na atrybuty z wieloma kategoriami.

Oto kilka zalet drzew decyzyjnych:

- Są zrozumiałe i łatwo interpretowalne, nawet dla osób niebędących ekspertami.
- Obsługują zarówno dane kategoryczne, jak i liczbowe.
- Radzą sobie z brakującymi danymi i wartościami odstającymi.
- Można używać ich do selekcji cech.
- Można łączyć je z innymi modelami w rozwiązania zespołowe, takie jak drzewa losowe.

Oto kilka wad drzew decyzyjnych:

- Bywają podatne na nadmierne dopasowanie, zwłaszcza jeśli drzewo jest zbyt głębokie lub złożone.
- Bywają wrażliwe na małe zmiany w danych albo sposób, w jaki budowane jest drzewo.

- Bywają ukierunkowane na cechy z wieloma kategoriami albo wysoką kardynalnością.
- Miewają problemy z rzadkimi zdarzeniami i niezrównoważonymi zbiorami danych.

Las losowy

Las losowy to wszechstronna, zespołowa metoda uczenia maszynowego, którą można wykorzystać w zadaniach klasyfikacji lub regresji. Podczas treningu generowanych jest wiele drzew decyzyjnych, po czym w zadaniach klasyfikacji klasę docelową przewiduje się na podstawie prognoz większości drzew, a w zadaniach regresji wartością przewidywaną jest średnia prognoz drzew. Algorytm budowania lasu losowego można podsumować następująco:

1. **Próbkowanie bootstrapowe**. Losowo wybierz podzbiór danych z zastępowaniem wartości, aby uzyskać nowy zbiór danych o tym samym rozmiarze co pierwotny zbiór danych.
2. **Selekcja cech**. Losowo wybierz podzbiór cech (kolumn) przy każdym podziale podczas budowania drzewa decyzyjnego. Pomaga to zróżnicować drzewa i ograniczyć nadmierne dopasowywanie.
3. **Budowanie drzewa**. Skonstruuj drzewo decyzyjne dla każdej próbki boostrapowej i podzbioru cech. Drzewa decyzyjne buduje się rekurencyjnie, dzieląc dane na podstawie wybranych cech aż do spełnienia kryterium zatrzymania (takiego jak maksymalna głębokość drzewa albo minimalna liczba próbek w węźle liściowym).
4. **Uczenie zespołowe**. Połącz prognozy wszystkich drzew decyzyjnych w celu dokonania końcowej prognozy. W klasyfikacji ostateczną prognozą jest klasa, która otrzyma najwięcej głosów od drzew decyzyjnych. W regresji ostateczną prognozą jest średnia prognoz wszystkich drzew decyzyjnych.

Algorytm lasu losowego można sformułować matematycznie w opisany niżej sposób.

Na podstawie zbioru danych D zawierającego N próbek i M cech tworzymy T drzew decyzyjnych $\{Drzewo_1, Drzewo_2, \ldots, Drzewo_T\}$ z użyciem opisanych wyżej czynności. Każde drzewo jest konstruowane z użyciem bootstrapowej próbki danych D' o rozmiarze N' ($N' <= N$) i podzbioru cech F' o rozmiarze m ($m <= M$). Dla każdego podziału w drzewie decyzyjnym losowo wybieramy k ($k < m$) cech z F' i wybieramy najlepszą cechę do podziału danych na podstawie miary nieczystości (np. współczynnika Giniego albo entropii). Drzewo decyzyjne jest budowane aż do osiągnięcia kryterium zatrzymania (takiego jak maksymalna głębokość drzewa albo minimalna liczba próbek w węźle liściowym).

Ostateczną prognozę $\hat{\mathbf{y}}$ dla nowej próbki x uzyskuje się przez agregowanie prognoz wszystkich drzew decyzyjnych.

W przypadku klasyfikacji $\hat{\mathbf{y}}$ jest klasą, która otrzymała najwięcej głosów od wszystkich drzew decyzyjnych:

$$\hat{y} = \text{argmax}_j \sum_i I(y_{i,j} = 1)$$

W tym równaniu y_{ij} jest prognozą *j*-tego drzewa dla *i*-tej próbki, a *I*() jest funkcją wskaźnikową, która zwraca 1, kiedy warunek jest spełniony, a 0 w przeciwnym przypadku.

W przypadku regresji $\hat{\mathbf{y}}$ jest średnią prognoz wszystkich drzew decyzyjnych:

$$\hat{y} = \left(\frac{1}{T}\right) \sum_{i=1}^{T} y_i$$

W tym równaniu y_i jest prognozą *i*-tego drzewa decyzyjnego dla nowej próbki *x*.

Podsumowując, las losowy to użyteczny algorytm uczenia maszynowego, który dobrze radzi sobie z wysokowymiarowymi i zaszumionymi zbiorami danych. Buduje wiele drzew decyzyjnych z użyciem bootstrapowych próbek danych i podzbiorów cech, a następnie agreguje przewidywania wszystkich drzew decyzyjnych w celu dokonania ostatecznej prognozy. Algorytm jest skalowalny, łatwy w użyciu i zapewnia miarę ważności cech, dlatego jest popularną opcją w wielu zastosowaniach uczenia maszynowego.

Zalety lasów losowych są następujące:

- **Niezawodność**. Las losowy to bardzo stabilny algorytm, który radzi sobie z różnymi typami danych wejściowych, m.in. liczbowymi, kategorycznymi i porządkowymi.
- **Selekcja cech**. Lasy losowe mogą szeregować cechy według ważności, co pozwala użytkownikowi wybrać najważniejsze cechy do zadań klasyfikacji lub regresji.
- **Nadmierne dopasowanie**. Lasy losowe mają wbudowany mechanizm (tzw. bagging), który ogranicza nadmierne dopasowanie, dzięki czemu model dobrze generalizuje się na nowe dane.
- **Skalowalność**. Lasy losowe radzą sobie z dużymi zbiorami danych zawierającymi wiele funkcji, więc dobrze nadają się do aplikacji big data.
- **Wartości odstające**. Lasy losowe są odporne na obecność wartości odstających, ponieważ są oparte na drzewach decyzyjnych, które dobrze radzą sobie z wartościami odstającymi.

Wady lasów losowych są następujące:

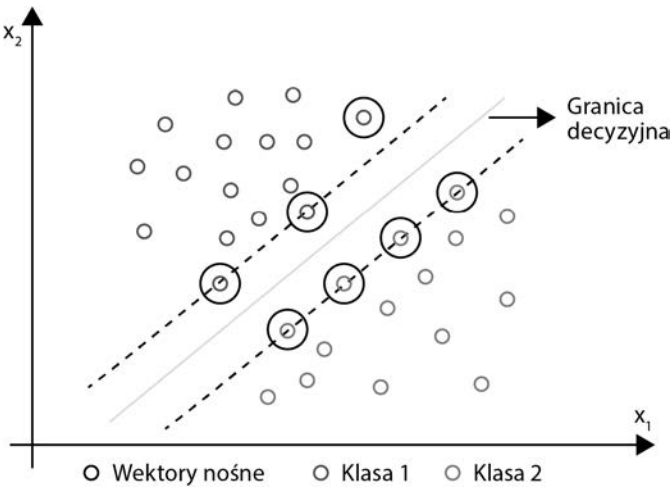
- **Interpretowalność**. Interpretowanie lasów losowych bywa trudne, ponieważ są one oparte na zespole drzew decyzyjnych.
- **Czas treningu**. Czas treningu lasu losowego jest dłuższy niż w przypadku innych prostych algorytmów, zwłaszcza jeśli liczba drzew decyzyjnych w zespole jest duża.
- **Użycie pamięci**. Lasy losowe wymagają więcej pamięci niż niektóre inne algorytmy, ponieważ muszą przechowywać drzewa decyzyjne w pamięci.
- **Stronniczość**. Lasy losowe bywają stronnicze, jeśli dane są niezrównoważone albo zmienna docelowa ma wysoką kardynalność.

- **Nadmierne dopasowanie**. Choć las losowy jest zaprojektowany tak, aby zapobiegał nadmiernemu dopasowaniu, nadal można przeuczyć model, jeśli hiperparametry nie są poprawnie dostrojone.

Ogólnie rzecz biorąc, las losowy to zaawansowany algorytm uczenia maszynowego, który ma wiele zalet, ale przed zastosowaniem go do konkretnego problemu trzeba rozważyć jego ograniczenia.

Maszyny wektorów nośnych (SVM)

Maszyny wektorów nośnych (ang. *support vector machine*, **SVM**) to algorytmy uczenia nadzorowanego, które mogą wykonywać zarówno zadania klasyfikacji, jak i regresji. Szczególnie dobrze działają w scenariuszach ze skomplikowanymi granicami decyzyjnymi, przekraczając ograniczenia modeli liniowych. Celem maszyn SVM jest zasadniczo zidentyfikowanie hiperpłaszczyzny, która maksymalnie segreguje klasy w wielowymiarowej przestrzeni. Hiperpłaszczyznę sytuuje się tak, aby zmaksymalizować odległość między nią a najbliższymi punktami z każdej klasy, znanymi jako wektory nośne. Oto jak działają SVM w przypadku problemu klasyfikacji binarnej. Na podstawie zbioru danych treningowych $\{(x_1,y_1), (x_2, y_2), \ldots (x_n, y_n)\}$, gdzie x_i jest d-wymiarowym wektorem cech, a y_i jest binarną etykietą klasy (+1 lub –1), celem SVM jest znalezienie hiperpłaszczyzny, która rozdziela dwie klasy z największym marginesem. Margines definiuje się jako odległość między hiperpłaszczyzną a najbliższymi punktami danych każdej klasy (patrz rysunek 3.2).



Rysunek 3.2. Marginesy SVM

Hiperpłaszczyzna jest zdefiniowana przez wektor wagi w i wyraz biasu b tak, że dla dowolnego nowego punktu danych x przewidywana etykieta klasy y jest dana poniższym równaniem:

$$y = sign(\mathbf{w}^T\mathbf{x}+\mathrm{b})$$

W tym równaniu *sign* to funkcja znaku, która zwraca +1, jeśli argument jest dodatni, i –1 w przeciwnym przypadku.

Funkcją celu SVM jest minimalizacja błędu klasyfikacji przy zastrzeżeniu, że margines zostanie zmaksymalizowany. Można to sformułować jako problem optymalizacji:

$$zminimalizuj\ 1{:}2||\mathbf{w}||^2$$

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

W równaniu tym $||\mathbf{w}||^2$ jest podniesioną do kwadratu normą euklidesową wektora wagi *w*. Ograniczenia gwarantują, że wszystkie punkty danych będą poprawnie klasyfikowane, a margines zostanie zmaksymalizowany.

Oto kilka zalet maszyn SVM:

- Działają efektywnie w wysokowymiarowych przestrzeniach, co jest przydatne, kiedy liczba cech jest duża.
- Można używać ich zarówno do zadań klasyfikacji, jak i regresji.
- Działają dobrze zarówno z danymi rozdzielnymi liniowo, jak i nierozdzielnymi liniowo.
- Dzięki koncepcji marginesu radzą sobie z wartościami odstającymi.
- Mają parametr regularyzacji, który pozwala kontrolować nadmierne dopasowanie.

Oto kilka wad maszyn SVM:

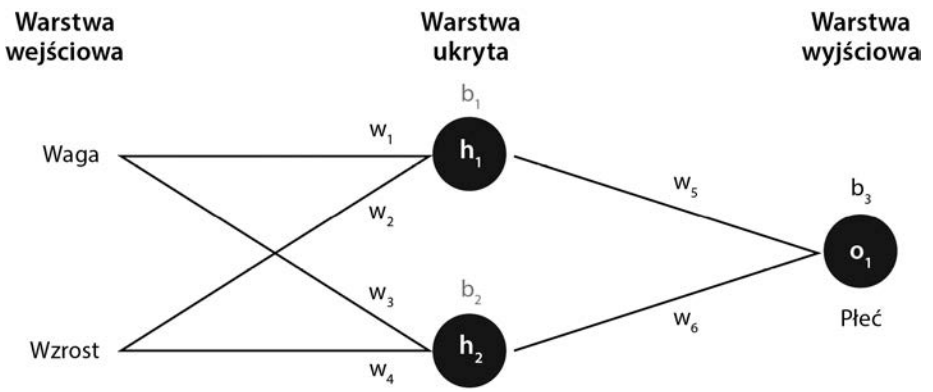
- Są wrażliwe na wybór funkcji jądrowej, która ma duży wpływ na działanie modelu.
- Są kosztowne obliczeniowo w przypadku dużych zbiorów danych.
- Interpretacja wyników modelu SVM jest trudna.
- Wymagają uważnego strojenia parametrów w celu osiągnięcia dobrych wyników.

Sieci neuronowe i transformery

Sieci neuronowe i transformery to zaawansowane modele uczenia maszynowego, których używa się do różnych zadań, takich jak klasyfikacja obrazu, NLP i rozpoznawanie mowy.

Sieci neuronowe

Inspiracją do powstania sieci neuronowych były struktura i funkcjonowanie ludzkiego mózgu. Reprezentują one kategorię modeli uczenia maszynowego, które biegle wykonują różnorodne zadania, takie jak klasyfikacja, regresja i nie tylko. Sieci neuronowe składają się z wielu warstw wzajemnie połączonych węzłów nazywanych neuronami. Sygnały wyjściowe każdej warstwy trafiają na wejście następnej warstwy, przez co powstaje hierarchia reprezentacji cech. Wejściem pierwszej warstwy są surowe dane, a wyjściem ostatniej warstwy jest prognoza. Prostą sieć neuronową do wykrywania płci osoby na podstawie jej wzrostu i wagi przedstawiono na rysunku 3.3.



Rysunek 3.3. Prosta sieć neuronowa

Działanie pojedynczego neuronu w sieci neuronowej można opisać następującym równaniem:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

W równaniu tym x_i to wartości wejściowe, w_i to wagi połączeń między neuronami, b to wyraz biasu (przesunięcia), a $f()$ to funkcja aktywacji. Funkcja aktywacji stosuje nieliniowe przekształcenie do ważonej sumy wejść i wyrazu biasu.

Trenowanie sieci neuronowej polega na regulowaniu wag i biasów neuronów w celu zminimalizowania funkcji straty. Zwykle robi się to z użyciem algorytmu optymalizacyjnego, takiego jak stochastyczne zejście gradientowe. Do zalet sieci neuronowych należą ich zdolność do uczenia się złożonych, nieliniowych relacji między danymi wejściowymi a wyjściowymi, zdolność do automatycznego wyodrębniania znaczących cech z surowych danych oraz skalowalność wymagana do przetwarzania dużych zbiorów danych.

Do wad sieci neuronowych należą wysokie wymagania w zakresie mocy obliczeniowej i pamięci, wrażliwość na hiperparametry oraz trudności z interpretowaniem ich wewnętrznych reprezentacji.

Transformery

Transformery to odmiana sieci neuronowej, która szczególnie dobrze nadaje się do przetwarzania danych sekwencyjnych, takich jak tekst lub mowa. Zostały one wprowadzone w kontekście NLP, a od tego czasu zastosowano je również do szerokiej gamy innych zadań.

Kluczowym komponentem transformera jest mechanizm samouwagi, który pozwala modelowi „zwracać uwagę" na różne części sekwencji wejściowej podczas obliczania wyjścia. Mechanizm samouwagi opiera się na iloczynie skalarnym między wektorem kwerendy, zbiorem wektorów kluczy i zbiorem wektorów wartości. Wynikowe wagi atencji służą do skalowania wartości, które następnie łączy się w celu utworzenia danych wyjściowych.

Operację samouwagi można przedstawić poniższymi równaniami:

$$Q = XW_Q$$

$$K = XW_K$$

$$V = XW_V$$

$$A(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_K}}\right)V$$

W równaniach tych X jest sekwencją wejściową, W_Q, W_K i W_V to wyuczone macierze rzutowania dla wektorów (odpowiednio) kwerendy, kluczy i wartości, d_K to wymiarowość wektorów kluczy.

Do zalet transformerów należą ich zdolność do przetwarzania sekwencji wejściowych różnej długości, zdolność do wychwytywania zależności między odległymi elementami danych oraz najwyższa obecnie skuteczność w wielu zadaniach NLP.

Do wad transformerów należą wysokie wymagania w zakresie mocy obliczeniowej i pamięci, wrażliwość na hiperparametry oraz trudności z modelowaniem zadań, które wymagają jawnego modelowania dynamiki sekwencyjnej.

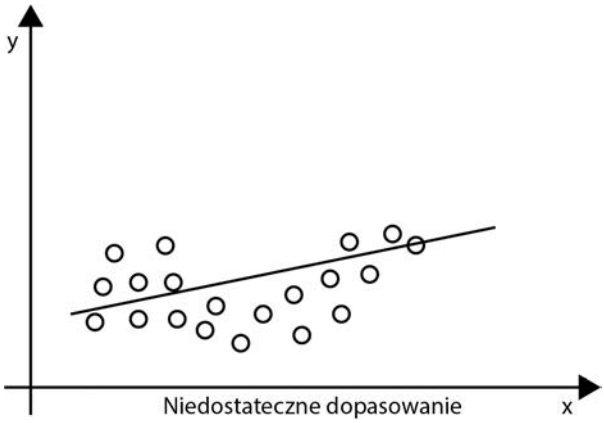
Powyżej opisano tylko kilka najpopularniejszych modeli uczenia maszynowego. Wybór modelu zależy od problemu, rozmiaru i jakości danych oraz pożądanego wyniku. Teraz gdy omówiliśmy najczęściej używane modele uczenia maszynowego, wyjaśnimy, czym jest niedostateczne lub nadmierne dopasowanie, do którego może dojść w procesie treningu.

Niedostateczne i nadmierne dopasowanie modelu

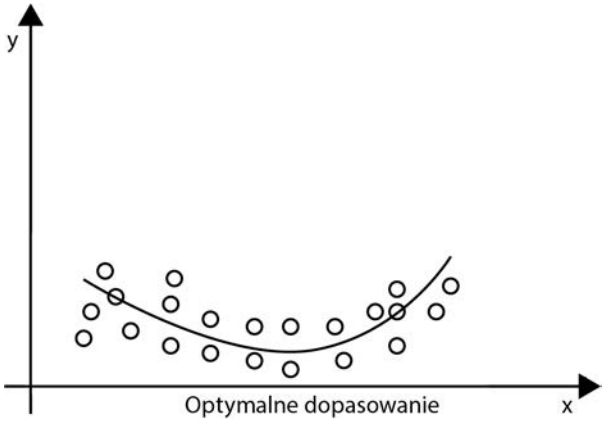
W uczeniu maszynowym ostatecznym celem jest zbudowanie modelu, który dobrze generalizuje się na niewidziane wcześniej dane. Czasem jednak model nie osiąga tego celu z powodu niedostatecznego albo nadmiernego dopasowania.

Do niedostatecznego dopasowania (zwanego też niedouczeniem) dochodzi wtedy, kiedy model jest zbyt prosty, żeby uchwycić wzorce w danych. Innymi słowy, model nie może poprawnie nauczyć się związków między cechami a zmienną docelową. Może to prowadzić do słabych wyników zarówno na danych treningowych, jak i testowych. Na przykład na rysunku 3.4 widać, że model jest niedostatecznie dopasowany i nie reprezentuje dobrze danych. W uczeniu maszynowym zazwyczaj preferujemy precyzyjne modele, takie jak pokazany na rysunku 3.5.

Niedostateczne dopasowanie może wynikać z niewłaściwego treningu albo zbyt małej złożoności modelu. Aby rozwiązać ten problem, możemy użyć bardziej złożonych modeli i kontynuować proces treningu.



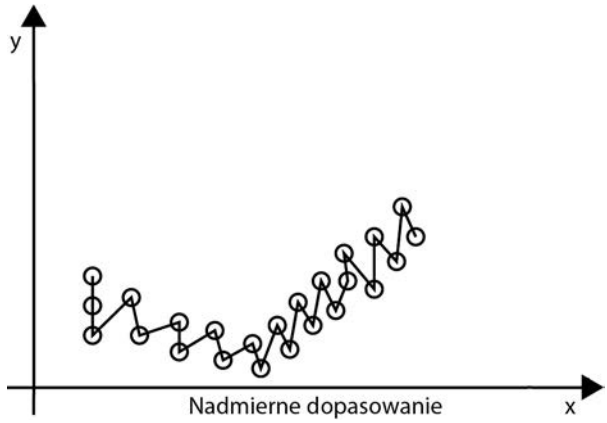
Rysunek 3.4. Model uczenia maszynowego niedostatecznie dopasowany do danych treningowych



Rysunek 3.5. Model uczenia maszynowego optymalnie dopasowany do danych treningowych

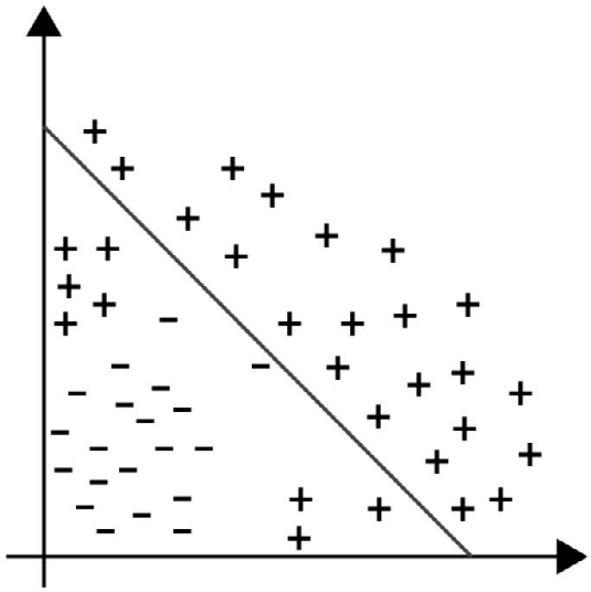
Optymalne dopasowanie ma miejsce wtedy, kiedy model dobrze odzwierciedla wzorce w danych, ale nie dopasowuje się do każdej indywidualnej próbki. Dzięki temu może działać lepiej na niewidzianych wcześniej danych.

Natomiast do nadmiernego dopasowania (zwanego też przeuczeniem) dochodzi wtedy, gdy model jest zbyt złożony i dopasowuje się do danych zbyt ściśle, przez co źle się generalizuje i nie działa dobrze na nowych, niewidzianych wcześniej danych, jak pokazano na rysunku 3.6. Nadmiernie dopasowany model uczy się szumu albo losowych fluktuacji w danych zamiast rzeczywistych wzorców. Innymi słowy, model jest zbyt wyspecjalizowany pod kątem danych treningowych i nie działa dobrze na danych testowych. Jak widać na rysunku, model próbuje bardzo precyzyjnie przewidzieć każdą próbkę. Problem w tym, że model nie uczy się ogólnego wzorca, a tylko rozkładu poszczególnych próbek, przez co osiąga słabe wyniki, kiedy otrzyma nowe rekordy.



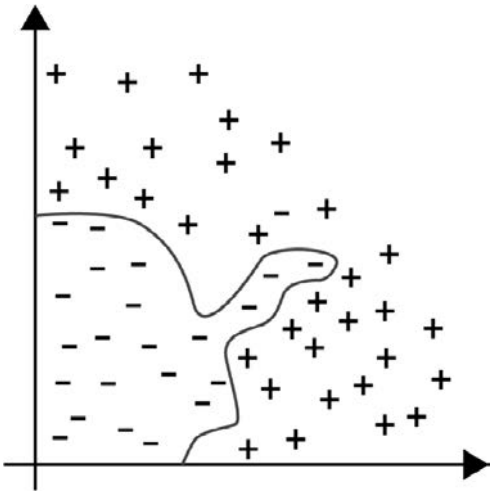
Rysunek 3.6. Model uczenia maszynowego optymalnie dopasowany do danych treningowych

Aby zrozumieć kompromis między niedostatecznym a nadmiernym dopasowaniem, warto rozważyć kompromis między obciążeniem a wariancją. Obciążenie (inaczej bias) to różnica między wartościami przewidywanymi przez model a rzeczywistymi wartościami w danych treningowych. Wysokie obciążenie oznacza, że model nie jest wystarczająco złożony, aby uchwycić wzorce w danych, więc nie może się do nich wystarczająco dopasować. Niedostatecznie dopasowany model działa słabo zarówno na danych treningowych, jak i testowych (rysunek 3.7).



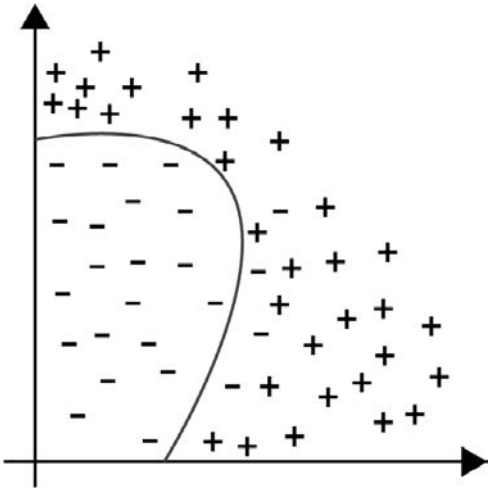
Rysunek 3.7. Wysokie obciążenie

Z kolei wariancja opisuje wrażliwość modelu na małe fluktuacje w danych. Wysoka wariancja oznacza, że model jest zbyt skomplikowany i nadmiernie dopasowuje się do danych, przez co źle się generalizuje i nie działa dobrze na nowych danych. Nadmiernie dopasowany model osiąga dobre wyniki na danych treningowych, ale złe na danych testowych (rysunek 3.8).



Rysunek 3.8. Wysoka wariancja

Aby osiągnąć równowagę między obciążeniem a wariancją, trzeba wybrać model, który nie jest ani zbyt prosty, ani zbyt złożony. Jak wspomniano wcześniej, często nazywa się to kompromisem między obciążeniem a wariancją (rysunek 3.9). Model z wysokim obciążeniem i niską wariancją można ulepszyć przez zwiększenie złożoności, a model z wysoką wariancją i niskim obciążeniem można ulepszyć przez zmniejszenie złożoności.



Rysunek 3.9. Model dokładnie taki, jak trzeba (bez wysokiego obciążenia, bez wysokiej wariancji)

Istnieje kilka metod ograniczania obciążenia i wariancji w modelu. Często używa się regularyzacji, która dodaje wyraz kary do funkcji straty w celu kontrolowania złożoności modelu. Innym podejściem jest użycie zespołów, które łączą wiele modeli w celu poprawienia ogólnych wyników przez ograniczenie wariancji. Można też skorzystać z walidacji krzyżowej, aby ocenić trafność modelu i dostroić jego hiperparametry w celu uzyskania optymalnej równowagi między obciążeniem a wariancją.

Ogólnie rzecz biorąc, zrozumienie obciążenia i wariancji w uczeniu maszynowym jest bardzo ważne, ponieważ pomaga wybrać odpowiedni model i zidentyfikować źródła błędów.

Obciążenie odnosi się do błędu, który powstaje przez przybliżenie rzeczywistego problemu z użyciem uproszczonego modelu. Wariancja odnosi się natomiast do błędu, który powstaje w wyniku wrażliwości modelu na małe fluktuacje w danych treningowych.

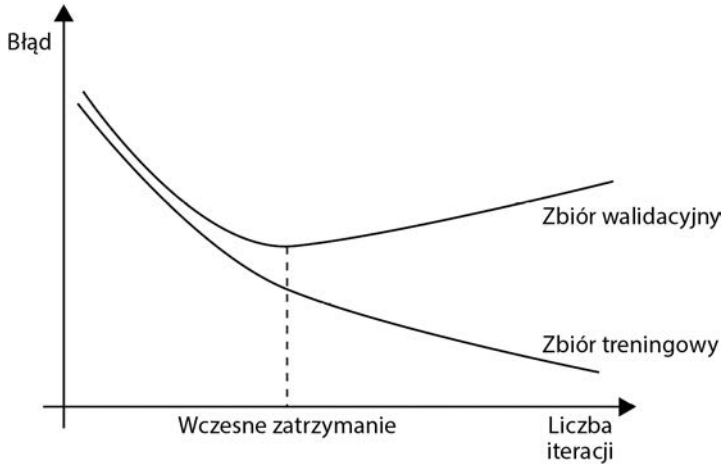
Kiedy model ma wysokie obciążenie i niską wariancję, jest niedostatecznie dopasowany (niedouczony). Oznacza to, że model nie odzwierciedla złożoności problemu i przyjmuje zbyt uproszczone założenia. Kiedy model ma niskie obciążenie i wysoką wariancję, jest nadmiernie dopasowany (przeuczony). Oznacza to, że jest zbyt wrażliwy na dane treningowe i dopasowuje się do szumu, zamiast do rzeczywistych wzorców.

Aby zapobiec niedostatecznemu dopasowaniu, można zwiększyć złożoność modelu, dodać więcej cech albo posłużyć się bardziej zaawansowanym algorytmem. Aby zapobiec nadmiernemu dopasowaniu, można skorzystać z kilku metod:

- **Walidacja krzyżowa**. Ocena działania modeli uczenia maszynowego ma kluczowe znaczenie. Walidacja krzyżowa to metoda oceny efektywności modelu uczenia maszynowego. Polega ona na trenowaniu modelu na jednej części danych i testowaniu go na drugiej. Przez wykorzystanie różnych podzbiorów danych do treningu i ewaluacji walidacja krzyżowa ogranicza ryzyko nadmiernego dopasowania. Technika ta zostanie opisana dokładniej w następnym podrozdziale poświęconym podziałowi danych.
- **Regularyzacja**. Regularyzacja to technika, która dodaje wyraz kary do funkcji straty podczas treningu, co pomaga ograniczyć złożoność modelu i zapobiec nadmiernemu dopasowaniu. Istnieją różne typy regularyzacji, w tym regularyzacja L1 (LASSO), regularyzacja L2 (grzbietowa) oraz regularyzacja sieci elastycznej.
- **Wczesne zatrzymanie**. Wczesne zatrzymanie polega na przerwaniu procesu treningowego, kiedy wyniki modelu na danych walidacyjnych zaczynają się pogarszać. Zapobiega to nadmiernemu dopasowaniu, powstrzymując model przed dalszym uczeniem się na danych treningowych, kiedy osiągnął on już maksymalną trafność. Techniki tej używa się zwykle w algorytmach iteracyjnych, takich jak metody uczenia głębokiego, w których model jest trenowany przez wiele cykli (tzw. epok). Aby z niej skorzystać, zwykle trenuje się model, jednocześnie oceniając jego działanie na podzbiorze treningowym i walidacyjnym.

  Model zwykle działa tym lepiej na zbiorze treningowym, im dłużej trwa trening, ale ponieważ nie widział wcześniej zbioru walidacyjnego, błąd walidacji zazwyczaj początkowo maleje, a w pewnym momencie zaczyna rosnąć. Jest to punkt, w którym model zaczyna nadmiernie dopasowywać się

do danych treningowych. Poprzez wizualizację błędu treningowego i walidacyjnego podczas treningu możemy zidentyfikować ten punkt i przerwać trening (rysunek 3.10).



Rysunek 3.10. Wczesne zatrzymanie

- **Porzucanie**. Porzucanie (ang. *dropout*) to technika używana w modelach uczenia głębokiego do losowego eliminowania niektórych neuronów podczas treningu (przez ustawienie ich wag na zero). Zapobiega to sytuacji, w której model zanadto skupia się na niewielkim podzbiorze cech, przez co nadmiernie dopasowuje się do danych. Zerując wagi neuronów podczas treningu, sprawiamy, że model uczy się ogólnych wzorców, zamiast po prostu zapamiętywać dane treningowe.
- **Wzbogacanie danych**. Wzbogacanie danych to metoda sztucznego powiększania treningowego zbioru danych przez stosowanie przekształceń, takich jak obracanie, skalowanie i przestawianie, do istniejących danych. Strategia ta ogranicza nadmierne dopasowanie, oferując modelowi bardziej zróżnicowany zbiór przykładów do nauki.
- **Metody zespołowe**. Metody zespołowe to techniki łączenia wielu modeli w celu zwiększenia efektywności i zapobiegania przeuczeniu. Przykładami takich technik są bagging, wzmacnianie (ang. *boosting*) i spiętrzanie (ang. *stacking*).

Techniki te pomagają zapobiegać nadmiernemu dopasowaniu i budować modele, które dobrze generalizują się na nowe, niewidziane wcześniej dane. W praktyce trzeba monitorować działanie modelu zarówno podczas treningu, jak i podczas testów, i dokonywać odpowiednich regulacji, aby uzyskać najlepszą możliwą generalizację. W następnym podrozdziale wyjaśnimy, jak podzielić dostępne dane na zbiór treningowy i testowy.

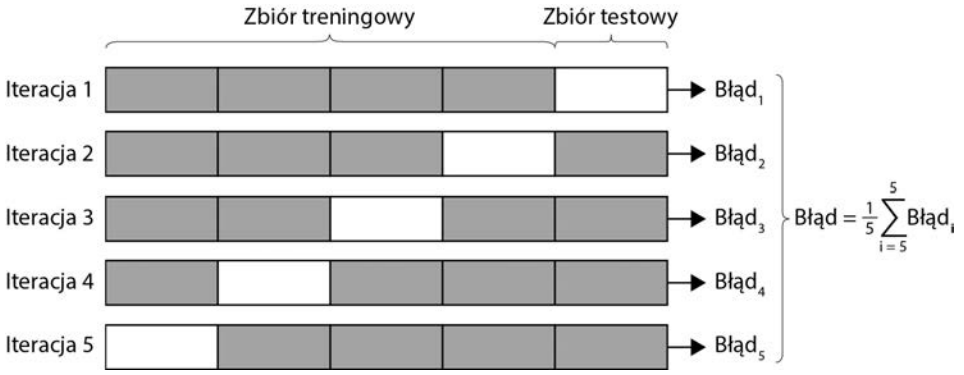
Dzielenie danych

Podczas tworzenia modelu uczenia maszynowego należy podzielić dane na treningowe, testowe i walidacyjne. Pozwala to ocenić działanie modelu na nowych, niewidzianych wcześniej danych i zapobiec nadmiernemu dopasowaniu.

Najpopularniejszą metodą dzielenia danych jest podział treningowo-testowy, który dzieli dane na dwa zbiory: zbiór treningowy służący do uczenia modelu i zbiór testowy służący do oceny efektywności modelu. Dane dzieli się losowo na dwa zbiory, przy czym zwykle 80% danych wykorzystuje się do treningu, a 20% do testów. Przy takim podejściu model uczy się na większości danych (zbiorze treningowym) i jest testowany na pozostałych (zbiorze testowym). Pozwala to zbadać działanie modelu na nowych, niewidzianych wcześniej danych.

Większość modeli uczenia maszynowego ma zbiór hiperparametrów, które można dostrajać (dostrajanie hiperparametrów zostanie opisane w następnym punkcie). Kiedy dostrajamy hiperparametry, chcielibyśmy wiedzieć, czy wyniki uzyskane na zbiorze testowym są wiarygodne, a nie wynikają tylko z przypadkowego doboru hiperparametrów. W takim przypadku, w zależności od rozmiaru danych treningowych, możemy podzielić dane w proporcjach 60%, 20%, 20% (albo 70%, 15%, 15%) na treningowe, walidacyjne i testowe. Uczymy model na danych treningowych i wybieramy zbiór hiperparametrów, który daje najlepsze wyniki na zbiorze walidacyjnym. Następnie raportujemy rzeczywiste wyniki modelu na zbiorze testowym, który nie był używany ani podczas treningu, ani podczas wyboru hiperparametrów.

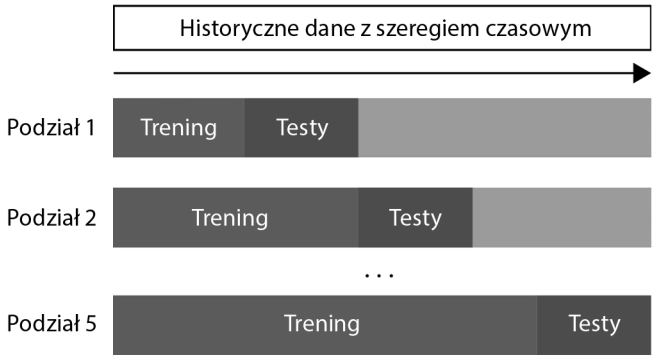
Bardziej zaawansowaną metodą dzielenia danych, przydatną zwłaszcza wtedy, kiedy rozmiar danych treningowych jest ograniczony, jest *k*-krotna walidacja krzyżowa. W metodzie tej dane dzieli się na *k* równych części nazywanych „krotnościami", a model jest trenowany i testowany *k* razy, przy czym każda krotność jest używana raz jako zbiór testowy, a pozostałe jako zbiór treningowy. Następnie wyniki dla każdej krotności uśrednia się, aby uzyskać ogólną miarę efektywności modelu. *K*-krotna walidacja krzyżowa jest użyteczna w przypadku małych zbiorów danych, ponieważ standardowe dzielenie ich na część treningową i testową mogłoby prowadzić do dużej wariancji w ewaluacji wyników. W metodzie tej raportujemy średnią, minimalną i maksymalną efektywność modelu na każdej z *k* krotności, jak pokazano na rysunku 3.11.



Rysunek 3.11. K-krotna walidacja krzyżowa

Innym wariantem *k*-krotnej walidacji krzyżowej jest stratyfikowana *k*-krotna walidacja krzyżowa, która gwarantuje jednakowy rozkład zmiennej docelowej we wszystkich krotnościach. Jest to przydatne w przypadku niezrównoważonych zbiorów danych, w których liczba wystąpień jednej klasy jest znacznie niższa od innych.

Szczególnej uwagi wymaga dzielenie szeregów czasowych. W takim przypadku zwykle używa się metody zwanej walidacją krzyżową szeregów czasowych, która zachowuje temporalną kolejność danych. W metodzie tej dane dzieli się na wiele segmentów, z których każdy reprezentuje stały interwał czasowy. Model następnie trenuje się na przeszłych danych i testuje na przyszłych. Pomaga to ocenić skuteczność modelu w rzeczywistych scenariuszach. Przykład dzielenia danych z szeregiem czasowym pokazano na rysunku 3.12.



Rysunek 3.12. Dzielenie szeregów czasowych

We wszystkich przypadkach należy zagwarantować, że podział będzie dokonywany losowo, ale za każdym razem z tym samym ziarnem losowym, aby zagwarantować odtwarzalność wyników. Trzeba też upewnić się, że podział jest reprezentatywny dla danych, tzn. rozkład zmiennej docelowej powinien być jednakowy we wszystkich zbiorach. Kiedy podzielimy dane na różne podzbiory do trenowania i testowania modelu, możemy spróbować znaleźć optymalny zbiór hiperparametrów modelu. Proces ten nazywa się dostrajaniem hiperparametrów i zostanie opisany w następnym podrozdziale.

Dostrajanie hiperparametrów

Dostrajanie hiperparametrów to ważny etap w procesie uczenia maszynowego, polegający na wybieraniu najlepszego zbioru hiperparametrów danego modelu. Hiperparametry to wartości, które ustawia się przed rozpoczęciem treningu i które mogą mieć znaczny wpływ na działanie modelu. Przykładami hiperparametrów mogą być tempo nauki, siła regularyzacji, liczba warstw ukrytych w sieci neuronowej i wiele innych.

Proces dostrajania hiperparametrów obejmuje wybieranie najlepszej kombinacji hiperparametrów, która zapewnia optymalne działanie modelu. Zwykle przeszukuje się w tym celu zestaw wstępnie zdefiniowanych hiperparametrów i ocenia ich działanie na zbiorze walidacyjnym.

Istnieje kilka metod dostrajania hiperparametrów, w tym wyszukiwanie siatkowe, wyszukiwanie losowe i optymalizacja bayesowska. Wyszukiwanie siatkowe polega na utworzeniu siatki wszystkich możliwych kombinacji hiperparametrów i ewaluowaniu każdej z nich na zbiorze walidacyjnym w celu określenia optymalnego zbioru hiperparametrów. Natomiast wyszukiwanie siatkowe losowo wybiera hiperparametry ze wstępnie zdefiniowanego rozkładu i ewaluuje ich działanie na zbiorze walidacyjnym.

Wyszukiwanie losowe i **siatkowe** to metody, które badają przestrzeń wyszukiwania, w całości lub losowo, bez uwzględniania poprzednich wyników. Są zatem nieefektywne. Zaproponowano alternatywną metodę optymalizacji bayesowskiej, która iteracyjnie oblicza rozkład *a posteriori* funkcji i uwzględnia przeszłe ewaluacje, aby znaleźć najlepsze hiperparametry. Używając tej metody, możemy znaleźć optymalny zestaw hiperparametrów przy mniejszej liczbie iteracji.

Optymalizacja bayesowska wykorzystuje przeszłe ewaluacje, aby probabilistycznie odwzorować hiperparametry na wyniki funkcji celu, jak pokazuje poniższe równanie:

$$P(wynik|hiperparametry)$$

Oto etapy optymalizacji bayesowskiej:

1. Opracowuje się substytucyjny model probabilistyczny dla funkcji celu.
2. Identyfikuje się optymalne hiperparametry na podstawie substytutu.
3. Wykorzystuje się te hiperparametry w rzeczywistej funkcji celu.
4. Aktualizuje się model substytucyjny w celu zintegrowania najnowszych wyników.
5. Powtarza się etapy od 2. do 4. do osiągnięcia maksymalnej liczby iteracji albo limitu czasu.

Metody **optymalizacji sekwencyjnej opartej na modelu** (ang. *sequential model-based optimization*, **SMBO**) to formalizacja optymalizacji bayesowskiej. Próby wykonuje się jedna po drugiej, za każdym razem wypróbowując lepsze hiperparametry i aktualizując model prawdopodobieństwa (substytut). Metody SMBO różnią się w etapach 3. i 4. — konkretnie sposobem, w jaki budują substytut funkcji celu, oraz kryteriami używanymi do wyboru następnych hiperparametrów. Do wariantów tych należą m.in. procesy gaussowskie, regresje z lasem losowym oraz drzewiaste estymatory Parzena.

W problemach niskowymiarowych z hiperparametrami liczbowymi optymalizacja bayesowska jest uważana za najlepszą dostępną metodę optymalizacji hiperparametrów. Jest ona jednak ograniczona do problemów o umiarkowanej liczbie wymiarów.

Oprócz opisanych wyżej metod dostępnych jest również kilka bibliotek, które automatyzują proces dostrajania hiperparametrów. Przykładem mogą być biblioteki `GridSearchCV` i `RandomizedSearchCV` z pakietu scikit-learn, `Keras Tuner` i `Optuna`. Biblioteki te umożliwiają efektywne dostrajanie parametrów i mogą znacznie poprawić działanie modeli uczenia maszynowego.

Optymalizacja hiperparametrów w uczeniu maszynowym może być złożonym i czasochłonnym procesem. Proces wyszukiwania wiąże się z dwoma głównymi wyzwaniami: czasem wykonania próby oraz złożonością przestrzeni wyszukiwania, w tym liczbą ocenianych kombinacji hiperparametrów. Problemy te są szczególnie widoczne w uczeniu

głębokim ze względu na dużą przestrzeń wyszukiwania i wykorzystanie dużych zbiorów treningowych.

Aby rozwiązać te problemy i ograniczyć przestrzeń wyszukiwania, można posłużyć się kilkoma standardowymi technikami. Na przykład zmniejszenie rozmiaru zbioru treningowego z użyciem próbkowania statystycznego albo zastosowanie technik selekcji cech może skrócić czas wykonania każdej próby. Ponadto identyfikowanie hiperparametrów najważniejszych z punktu widzenia optymalizacji oraz użycie funkcji celu innych niż dokładność, takich jak liczba operacji albo czas optymalizacji, może ograniczyć złożoność przestrzeni wyszukiwania.

Łącząc dokładność z wizualizacją z użyciem sieci dekonwolucyjnej, badacze zdołali uzyskać bardzo dobre wyniki. Trzeba jednak zaznaczyć, że techniki te nie są wyczerpujące, a najlepsze podejście może zależeć od konkretnego problemu.

Działanie modelu można też poprawić poprzez użycie wielu modeli działających równolegle; są to tak zwane modele zespołowe. Są one bardzo przydatne w rozwiązywaniu wielu problemów uczenia maszynowego.

Modele zespołowe

Modelowanie zespołowe to technika uczenia maszynowego, która łączy przewidywania wielu modeli, aby poprawić ogólne wyniki. Modele zespołowe opierają się na założeniu, że wiele modeli może działać lepiej niż jeden, ponieważ każdy z nich może uchwycić inne wzorce w danych.

Istnieje kilka typów modeli zespołowych, które omówimy w kolejnych punktach.

Bagging

Agregacja bootstrapowa, nazywana w skrócie **baggingiem** (od ang. *bootstrap aggregating*), to metoda, która łączy wiele niezależnych modeli wytrenowanych na innych podzbiorach danych treningowych w celu ograniczenia wariancji i poprawienia generalizacji.

Algorytm baggingu można podsumować w następujący sposób:

1. Na podstawie zbioru treningowego rozmiaru *n* utwórz *m* próbek bootstrapowych rozmiaru *n* (tzn. *m* razy wybierz *n* instancji poprzez próbkowanie z zastępowaniem).
2. Niezależnie wytrenuj model podstawowy (np. drzewo decyzyjne) na każdej próbce bootstrapowej.
3. Zagreguj przewidywania wszystkich modeli podstawowych, aby uzyskać prognozę zespołową. Można to zrobić albo przez głosowanie większościowe (w przypadku klasyfikacji), albo przez uśrednianie wyników (w przypadku regresji).

Algorytm baggingu jest szczególnie efektywny, kiedy modele podstawowe są niestabilne (tzn. mają wysoką wariancję), jak drzewa decyzyjne, albo kiedy treningowy zbiór danych jest mały.

Równanie agregacji przewidywań modeli podstawowych zależy od typu problemu (klasyfikacja lub regresja). W przypadku klasyfikacji prognozę zespołową uzyskuje się przez głosowanie większościowe:

$$Y_{ensemble} = \text{argmax}_j \sum_i I(y_{i,j} = 1)$$

W tym równaniu $y_{i,j}$ jest prognozą i-tego modelu podstawowego dla j-tej instancji, a $I()$ jest funkcją wskaźnikową, która zwraca 1, kiedy warunek jest spełniony, a 0 w przeciwnym przypadku.

W przypadku regresji prognoza zespołowa jest średnią prognoz poszczególnych modeli:

$$Y_{ensemble} = \left(\frac{1}{T}\right)\sum_{i=1}^{T} y_i$$

W tym równaniu y_i jest wartością przewidzianą przez i-ty model podstawowy.

Zalety baggingu są następujące:

- Lepsza generalizacja dzięki ograniczeniu wariancji i nadmiernego dopasowywania.
- Zdolność do obsługi wysokowymiarowych zbiorów danych ze złożonymi relacjami.
- Możliwość wykorzystania wielu różnych modeli podstawowych.

Wady baggingu są następujące:

- Większa złożoność modelu i dłuższy czas obliczeń ze względu na użycie wielu modeli podstawowych.
- Możliwość przeuczenia modelu zespołowego, jeśli modele podstawowe są zbyt złożone albo zbiór danych jest za mały.
- Bagging nie działa dobrze, kiedy modele podstawowe są ściśle skorelowane lub stronnicze.

Wzmacnianie

Wzmacnianie (ang. *boosting*) to inna popularna technika uczenia zespołowego, która ma na celu poprawienie wyników słabych klasyfikatorów poprzez połączenie ich w mocniejszy klasyfikator. Inaczej niż bagging, wzmacnianie skupia się na iteracyjnym poprawianiu dokładności klasyfikatora przez regulowanie wagi przykładów treningowych. Podstawowa idea jest taka, żeby uczyć się na pomyłkach poprzednich słabych klasyfikatorów i kłaść nacisk na przykłady, które zostały niepoprawnie sklasyfikowane w poprzednich iteracjach.

Istnieje kilka algorytmów wzmacniania, ale jednym z najpopularniejszych jest AdaBoost (skrót od ang. *adaptive boosting* — wzmacnianie adaptacyjne). Algorytm AdaBoost działa w następujący sposób:

1. Najpierw inicjalizuje się wagi przykładów treningowych tak, aby były równe.
2. Następnie trenuje się słaby klasyfikator na zbiorze treningowym.
3. Oblicza się ważony wskaźnik błędu słabego klasyfikatora.
4. Oblicza się ważność słabego klasyfikatora na podstawie ważonego wskaźnika błędu.
5. Zwiększa się wagi przykładów, które zostały źle sklasyfikowane przez słaby klasyfikator.
6. Normalizuje się wagi przykładów tak, aby sumowały się do jedności.
7. Powtarza się etapy od 2. do 6. do osiągnięcia wstępnie określonej liczby iteracji albo uzyskania pożądanej dokładności.
8. Na koniec łączy się słabe klasyfikatory w mocny klasyfikator poprzez przypisanie im wag zależnych od ich ważności.

Ostateczny klasyfikator jest ważoną kombinacją słabych klasyfikatorów. Ważność każdego słabego klasyfikatora zależy od jego ważonego wskaźnika błędu, który oblicza się z równania:

$$ERROR_m = \frac{\sum_{i=1}^{N} w_i I(y_i - h_m(x_i))}{\sum_{i=1}^{N} w_i}$$

W równaniu tym m jest indeksem słabego klasyfikatora, N jest liczbą przykładów treningowych, w_i jest wagą i-tego przykładu treningowego, y_i jest rzeczywistą etykietą i-tego przykładu treningowego, $h_m(x_i)$ jest prognozą m-tego słabego klasyfikatora dla i-tego przykładu treningowego, a $I(y_i–h_m(x_i))$ jest funkcją wskaźnikową, która zwraca 1, jeśli prognoza słabego klasyfikatora jest niepoprawna, a 0 w przeciwnym przypadku.

Ważność słabego klasyfikatora oblicza się z równania:

$$\alpha_m = \ln \frac{1 - error_m}{error_m}$$

Wagi przykładów aktualizuje się na podstawie ich ważności:

$$w_i = w_i^{\alpha_m I(y_i - h_m(x_i))}$$

Ostateczny klasyfikator uzyskuje się przez połączenie słabych klasyfikatorów:

$$H_x = sign\left(\sum_{m=1}^{M} \alpha_m h_m(x)\right)$$

W równaniu tym M jest łączną liczbą słabych klasyfikatorów, $h_m(x)$ jest prognozą m-tego słabego klasyfikatora, a *sign*() jest funkcją, która zwraca +1, jeśli jej argument jest dodatni, a –1 w przeciwnym przypadku.

Przyjrzyjmy się niektórym zaletom wzmacniania:

- Wzmacnianie może poprawić dokładność słabych klasyfikatorów i zapewnić znaczny wzrost trafności prognoz.
- Wzmacnianie jest względnie łatwe w implementacji i można stosować je w szerokiej gamie problemów klasyfikacji.
- Wzmacnianie radzi sobie z zaszumionymi danymi i ogranicza ryzyko nadmiernego dopasowania.

Oto niektóre wady wzmacniania:

- Wzmacnianie może być wrażliwe na wartości odstające i nadmiernie dopasowywać się do zaszumionych danych.
- Wzmacnianie może być kosztowne obliczeniowo, zwłaszcza w przypadku dużych zbiorów danych.
- Wzmacnianie może być trudne w interpretacji, ponieważ polega na łączeniu wielu słabych klasyfikatorów.

Spiętrzanie

Kolejną popularną metodą uczenia zespołowego jest układanie w stos albo spiętrzanie (ang. *stacking*), które wykorzystuje prognozy wielu modeli podstawowych do trenowania modelu wyższego poziomu. Idea polega na tym, aby wykorzystać mocne strony różnych modeli podstawowych w celu uzyskania wyższej efektywności predykcyjnej.

Oto jak działa spiętrzanie:

1. Dane treningowe dzieli się na dwie części. Pierwsza część służy do trenowania modeli podstawowych, a druga do tworzenia nowego zbioru z prognozami modeli podstawowych.
2. Na pierwszej części danych treningowych trenuje się wiele modeli podstawowych.
3. Wytrenowanych modeli podstawowych używa się do dokonania prognoz na drugiej części danych treningowych w celu utworzenia nowego zbioru danych z prognozami.
4. Na nowym zbiorze danych z prognozami trenuje się model wyższego poziomu (nazywany też metamodelem).
5. Wytrenowanego modelu wyższego poziomu używa się do dokonywania prognoz na zbiorze testowym.

Model wyższego poziomu jest zwykle prostym modelem, takim jak regresja liniowa, regresja logistyczna albo drzewo decyzyjne. Chodzi o to, aby wykorzystać prognozy modeli podstawowych jako cechy wejściowe modelu wyższego poziomu. W ten sposób model wyższego poziomu uczy się łączyć przewidywania modeli podstawowych w celu dokonywania dokładniejszych prognoz.

Lasy losowe

Jednym z najlepiej znanych modeli zespołowych jest las losowy, który łączy przewidywania wielu drzew decyzyjnych. Zwykle jest on dokładniejszy i mniej podatny na przeuczenie. Lasy losowe opisaliśmy wcześniej w tym rozdziale.

Wzmacnianie gradientowe

Wzmacnianie gradientowe (ang. *gradient boosting*) to kolejny model zespołowy, który można wykorzystać w zadaniach klasyfikacji i regresji. Zaczyna on od słabego klasyfikatora (takiego jak proste drzewo) i krok po kroku próbuje ulepszyć go, aby uzyskać skuteczniejszy model. Główna idea jest taka, że w każdym kroku model próbuje skupić się na swoich pomyłkach i poprawić swoje wyniki przez skorygowanie błędów w poprzednich drzewach.

W każdej iteracji algorytm oblicza ujemny gradient funkcji straty dla przewidzianych wartości, po czym dopasowuje drzewo decyzyjne do tych ujemnych wartości gradientu. Przewidywania nowych drzew łączy się następnie z przewidywaniami poprzednich drzew z użyciem parametru tempa nauki, który określa wkład każdego drzewa w końcową prognozę.

Ogólną prognozę modelu wzmacniania gradientowego uzyskuje się przez zsumowanie prognoz wszystkich drzew ważonych przez ich odpowiednie tempa nauki.

Przyjrzyjmy się równaniu używanemu przez algorytm wzmacniania gradientowego.

Najpierw inicjalizujemy model stałą wartością:

$$F_0(x) = \text{argmin}_c \sum_{i=1}^{N} L(y_i, c)$$

W równaniu tym c jest stałą, y_i jest rzeczywistą etykietą i-tej próbki, N jest liczbą próbek, a L jest funkcją straty, która mierzy błąd między etykietami przewidzianymi a rzeczywistymi.

W każdej iteracji m algorytm dopasowuje drzewo decyzyjne do ujemnych wartości gradientu funkcji straty dla przewidzianych wartości, $r_m = -\nabla L(y, F(x))$. Drzewo decyzyjne przewiduje ujemne wartości gradientu, których następnie używa się do zaktualizowania prognoz modelu zgodnie z poniższym równaniem:

$$F_m(x) = F_{m-1}(x) + \eta h_m(x)$$

W tym równaniu $F_{m-1}(x)$ to prognoza modelu w poprzedniej iteracji, η to tempo nauki, a $h_m(x)$ to prognoza drzewa decyzyjnego w bieżącej iteracji.

Ostateczną prognozę modelu uzyskuje się przez połączenie prognoz wszystkich drzew:

$$F(x) = \sum_{m=1}^{M} \eta_m h_m(x)$$

W tym równaniu M to łączna liczba drzew w modelu, a η_m i $h_m(x)$ to odpowiednio tempo nauki oraz prognoza m-tego drzewa.

Przyjrzyjmy się niektórym zaletom wzmacniania gradientowego:

- Zapewnia wysoką dokładność prognoz.
- Można wykorzystać je w problemach regresji i klasyfikacji.
- Radzi sobie z brakującymi danymi i wartościami odstającymi.
- Można używać go w połączeniu z różnymi funkcjami straty.
- Radzi sobie z danymi wysokowymiarowymi.

Teraz przyjrzyjmy się niektórym wadom:

- Podatne na przeuczenie, zwłaszcza kiedy liczba drzew jest duża.
- Trening jest kosztowny obliczeniowo i czasochłonny, zwłaszcza w przypadku dużych zbiorów danych.
- Wymaga uważnego dostrojenia hiperparametrów, takich jak liczba drzew, tempo nauki i maksymalna głębokość drzew.

Omówiliśmy rozwiązania zespołowe, które mogą poprawić działanie naszych modeli. Czasem jednak zbiór danych ma pewne właściwości, które musimy rozważyć, zanim zastosujemy modele uczenia maszynowego. Jednym z takich typowych przypadków jest niezrównoważony zbiór danych.

Dane niezrównoważone

W większości rzeczywistych problemów dane są niezrównoważone, co oznacza, że rozkład rekordów z różnych klas (na przykład pacjentów mających i niemających raka) jest różny. Obsługa niezrównoważonych zbiorów danych ma duże znaczenie w uczeniu maszynowym, ponieważ często spotyka się zbiory danych z nierównomiernym rozkładem klas. W takich przypadkach klasa mniejszościowa często jest niedostatecznie reprezentowana, co może prowadzić do niskiej trafności modelu i tendencyjnych prognoz. Powodem jest to, że metody uczenia maszynowego próbują zoptymalizować funkcję dopasowania, aby zminimalizować błąd w zbiorze treningowym. Przypuśćmy teraz, że 99% danych należy do klasy pozytywnej, a 1% do klasy negatywnej. W takim przypadku, jeśli model przewidzi wszystkie rekordy jako pozytywne, błąd wyniesie tylko 1%, jednak taki model będzie bezużyteczny. Kiedy więc mamy niezrównoważony zbiór danych, musimy sięgnąć po różne metody rozwiązywania tego problemu. Ogólnie rzecz biorąc, metody obsługi niezrównoważonych zbiorów danych dzielą się na trzy kategorie:

- **Podpróbkowanie** (ang. *undersampling*). Bardzo prostą metodą jest użycie mniejszej liczby rekordów z klasy większościowej. Metoda ta działa, ale trzeba pamiętać, że mniejsza ilość danych treningowych to mniej informacji przekazywanych do modelu, co przekłada się na niższą efektywność treningu i końcowego modelu.
- **Ponowne próbkowanie** (ang. *resampling*). Metody ponownego próbkowania modyfikują pierwotny zbiór danych w celu utworzenia zrównoważonego rozkładu. Można to osiągnąć albo przez nadpróbkowanie klasy mniejszościowej

(tworzenie większej liczby próbek klasy mniejszościowej), albo przez podpróbkowanie klasy większościowej (usuwanie próbek klasy większościowej). Do technik ponownego próbkowania należą: **nadpróbkowanie losowe**, **Synthetic Minority Oversampling Technique** (**SMOTE**) oraz **Adaptive Synthetic Sampling** (**ADASYN**). Do technik podpróbkowania należą: **podpróbkowanie losowe**, **łącza Tomeka** i **wyznaczanie centroidów klastrów**.

- **Obsługa niezrównoważonych zbiorów danych w modelach uczenia maszynowego**. Używa się takich rozwiązań jak zmodyfikowana funkcja kosztu albo zmodyfikowane dzielenie na partie w modelach uczenia głębokiego.

SMOTE

SMOTE to bardzo popularny algorytm do obsługi niezrównoważonych zbiorów danych w uczeniu maszynowym. Jest to technika generowania syntetycznych danych, która tworzy nowe, sztuczne próbki klasy mniejszościowej poprzez interpolację istniejących próbek. W tym celu SMOTE identyfikuje k najbliższych sąsiadów próbki klasy mniejszościowej, a następnie generuje nowe próbki wzdłuż odcinków łączących tych sąsiadów.

Oto etapy algorytmu SMOTE:

1. Wybierz próbkę klasy mniejszościowej, x.
2. Wybierz jednego z jej najbliższych sąsiadów, x'.
3. Wygeneruj syntetyczną próbkę przez interpolację między x a x'. W tym celu wybierz liczbę losową r z zakresu od 0 do 1 i oblicz syntetyczną próbkę w następujący sposób:

   $$\text{nowa próbka} = x + r(x' - x)$$

   W ten sposób powstaje nowa próbka, która leży gdzieś między próbkami x a x', ale różni się od każdej z nich.
4. Powtarzaj etapy od 1. do 3. aż do wygenerowania żądanej liczby syntetycznych próbek.

Oto zalety i wady metody SMOTE:

- Pomaga ona rozwiązać problem nierównowagi klasy przez tworzenie syntetycznych próbek klasy mniejszościowej.
- Można używać jej w połączeniu z innymi technikami, takimi jak podpróbkowanie losowe albo łącza Tomeka, aby jeszcze lepiej zrównoważyć zbiór danych.
- Można stosować ją zarówno na danych kategorycznych, jak i liczbowych.
- SMOTE czasem tworzy syntetyczne próbki, które są nierealistyczne lub zaszumione, co prowadzi do nadmiernego dopasowania.
- SMOTE sprawia czasem, że granica decyzyjna staje się zbyt wrażliwa na klasę mniejszościową, co prowadzi do gorszych wyników dla klasy większościowej.
- Metoda ta jest kosztowna obliczeniowo w przypadku dużych zbiorów danych.

Oto przykład zastosowania algorytmu SMOTE. Przypuśćmy, że mamy zbiór danych z dwiema klasami: klasa większościowa (klasa 0) ma 900 próbek, a klasa mniejszościowa (klasa 1) ma 100 próbek. Chcemy użyć metody SMOTE, aby wygenerować syntetyczne próbki klasy mniejszościowej:

1. Wybieramy próbkę klasy mniejszościowej, x.
2. Wybieramy jednego z jej najbliższych sąsiadów, x'.
3. Generujemy próbkę syntetyczną przez interpolację między x a x' z użyciem liczby losowej r:

   $$\text{nowa próbka} = x+r(x'-x)$$

   Przypuśćmy na przykład, że x to (1, 2), x' to (3, 4), a r to 0,5. W takim przypadku nowa próbka to:

   $$\text{nowa próbka} = (1,2)+0,5((3,4)-(1,2)) = (2,3)$$

4. Powtarzamy etapy od 1. do 3. aż do wygenerowania żądanej liczby syntetycznych próbek. Przypuśćmy, że chcemy wygenerować 100 syntetycznych próbek. Powtarzamy etapy od 1. do 3. dla każdej ze 100 próbek klasy mniejszościowej, a następnie łączymy pierwotne próbki klasy mniejszościowej z próbkami syntetycznymi i tworzymy zrównoważony zbiór danych, który zawiera po 200 próbek każdej klasy.

Algorytm NearMiss

Algorytm NearMiss to technika równoważenia rozkładu klas przez podpróbkowywanie (usuwanie) rekordów klasy większościowej. Kiedy dwie klasy mają rekordy, które są bardzo bliskie siebie, eliminacja niektórych rekordów klasy większościowej zwiększa odległość między dwiema klasami, co pomaga w procesie klasyfikacji. Algorytm NearMiss pomaga uniknąć problemu utraty informacji, który występuje w większości metod usuwania próbek.

Wyszukiwanie najbliższych sąsiadów składa się z następujących etapów:

1. Znajdź odległości między wszystkimi rekordami klasy większościowej i mniejszościowej. Celem jest usunięcie rekordów klasy większościowej.
2. Wybierz n rekordów klasy większościowej, które są najbliżej klasy mniejszościowej.
3. Jeśli w klasie mniejszościowej jest k rekordów, metoda najbliższych sąsiadów zwróci kn rekordów klasy większościowej.

Istnieją trzy odmiany stosowania algorytmu NearMiss, których możemy użyć do znalezienia n najbliższych rekordów klasy większościowej:

- Możemy wybrać rekordy klasy większościowej, których średnie odległości do k najbliższych rekordów klasy mniejszościowej są najmniejsze.
- Możemy wybrać rekordy klasy większościowej, których średnie odległości do k najdalszych rekordów klasy mniejszościowej są najmniejsze.

- Możemy zaimplementować dwa etapy. W pierwszym etapie zapisujemy *M* najbliższych sąsiadów każdego rekordu klasy mniejszościowej. Następnie wybieramy takie rekordy klasy większościowej, że ich średnia odległość do *N* najbliższych sąsiadów jest najmniejsza.

Uczenie wrażliwe na koszty

Uczenie wrażliwe na koszty to metoda używana do trenowania modeli uczenia maszynowego na niezrównoważonych zbiorach danych. W niezrównoważonych zbiorach danych liczba przykładów jednej klasy (klasy mniejszościowej) jest znacznie mniejsza niż liczba przykładów innej klasy (większościowej). Uczenie wrażliwe na koszty polega na przypisaniu modelowi kosztów błędnej klasyfikacji, które różnią się w zależności od przewidzianej klasy, co może pomóc modelowi skupić się bardziej na poprawnym klasyfikowaniu przykładów klasy mniejszościowej.

Załóżmy, że mamy problem klasyfikacji binarnej z dwiema klasami — pozytywną i negatywną. W uczeniu wrażliwym na koszty przypisujemy różne koszty różnym typom błędów. Możemy na przykład przypisać wyższy koszt błędnemu sklasyfikowaniu przykładu pozytywnego jako negatywny, ponieważ w niezrównoważonym zbiorze danych klasa pozytywna jest klasą mniejszościową, a błędne klasyfikowanie przykładów pozytywnych może mieć większy wpływ na działanie modelu.

Koszty możemy przypisać w formie macierzy błędów (tabela 3.2).

Tabela 3.2. Koszty w macierzy błędów

	Przewidziany pozytywny	Przewidziany negatywny
Rzeczywisty pozytywny	`Koszt_PP`	`Koszt_FN`
Rzeczywisty negatywny	`Kosz_FP`	`Koszt_PN`

`Koszt_PP`, `Koszt_FN`, `Koszt_FP` i `Koszt_PN` to koszty związane odpowiednio z klasyfikacjami prawdziwie pozytywnymi, fałszywie pozytywnymi, fałszywie negatywnymi i prawdziwie negatywnymi.

Aby wykorzystać macierz kosztów podczas treningu, możemy zmodyfikować standardową funkcję straty, którą model optymalizuje podczas treningu. Często używaną funkcją straty wrażliwej na koszty jest ważona strata entropii krzyżowej, zdefiniowana następująco:

$$strata = -(w_{pos} y \log(\hat{y}) + w_{neg}(1-y)\log(1-\hat{y}))$$

W tym równaniu y jest rzeczywistą etykietą (0 albo 1), $\hat{y}$ jest przewidzianym prawdopodobieństwem klasy pozytywnej, a w_{pos} i w_{neg} to wagi przypisane odpowiednio klasie pozytywnej i negatywnej.

Wagi w_{pos} i w_{neg} można określić na podstawie kosztów w macierzy błędów. Jeśli na przykład chcemy przypisać wyższy koszt klasyfikacjom fałszywie negatywnym (tzn. błędnemu klasyfikowaniu przykładów pozytywnych jako negatywne), możemy ustawić w_{pos} na wyższą wartość niż w_{neg}.

Uczenia wrażliwego na koszty można również używać w połączeniu z innymi typami modeli, takimi jak drzewa decyzyjne i maszyny wektorów nośnych. Koncepcję przypisywania różnych kosztów różnym typom błędów można stosować na różne sposoby w celu poprawienia działania modelu na niezrównoważonych zbiorach danych. Trzeba jednak uważnie wybrać odpowiednią macierz błędów i funkcję straty w zależności od charakterystyki zbioru danych i rozwiązywanego problemu:

- **Techniki zespołowe**. Techniki zespołowe łączą wiele modeli w celu poprawienia skuteczności predykcyjnej. W przypadku niezrównoważonych zbiorów danych zespół modeli można wytrenować na różnych podzbiorach zbioru danych, upewniając się, że każdy model jest trenowany zarówno na klasie mniejszościowej, jak i większościowej. Do technik zespołowych używanych do pracy z niezrównoważonymi zbiorami danych należy bagging i wzmacnianie.
- **Detekcja anomalii**. Techniki detekcji anomalii można wykorzystać do zidentyfikowania klasy mniejszościowej jako anomalii w zbiorze danych. Techniki te próbują zidentyfikować rzadkie zdarzenia, które różnią się znacznie od klasy większościowej. Zidentyfikowanych próbek można następnie użyć do wytrenowania modelu na klasie mniejszościowej.

Wzbogacanie danych

Wzbogacanie danych polega na generowaniu nowych przykładów poprzez przekształcanie istniejących z zachowaniem pierwotnej etykiety. Do przekształceń tych może należeć obracanie, przesuwanie, skalowanie, przestawianie i dodawanie szumu. Jest to szczególnie użyteczne w przypadku niezrównoważonych zbiorów danych, w których liczba przykładów jednej klasy jest znacznie mniejsza niż drugiej.

W kontekście niezrównoważonych zbiorów danych wzbogacania można użyć do tworzenia nowych przykładów klasy mniejszościowej, a w rezultacie do równoważenia zbioru danych. W tym celu można zastosować ten sam zbiór przekształceń do przykładów klasy mniejszościowej i uzyskać nowy zbiór przykładów, które nadal reprezentują klasę mniejszościową, ale nieco różnią się od pierwotnych.

Równania używane do wzbogacania danych są względnie proste, ponieważ opierają się na stosowaniu funkcji transformacyjnych do pierwotnych przykładów. Aby na przykład obrócić obraz o pewien kąt, możemy użyć macierzy obrotu:

$$x' = x\,cos(\theta) - y\,sin(\theta)$$

$$y' = x\,sin(\theta) + y\,cos(\theta)$$

W tym równaniu x i y to pierwotne współrzędne piksela obrazu, x' i y' to nowe współrzędne po obrocie, a θ to kąt obrotu.

Podobnie, aby zastosować przesunięcie, po prostu przesuwamy obraz o pewną liczbę pikseli:

$$x' = x + dx$$

$$y' = y + dy$$

W tym równaniu *dx* i *dy* oznaczają odpowiednio przesunięcie w poziomie i pionie.

Wzbogacanie danych to skuteczna technika radzenia sobie z niezrównoważonymi zbiorami danych, ponieważ pozwala utworzyć nowe przykłady reprezentujące klasę mniejszościową z zachowaniem etykiet. Trzeba jednak używać jej ostrożnie, ponieważ może również wprowadzać do danych szum i artefakty oraz prowadzić do nadmiernego dopasowania.

Podsumowując, obsługa niezrównoważonych zbiorów danych jest ważnym aspektem uczenia maszynowego. Istnieje kilka technik obsługi niezrównoważonych zbiorów danych, a każda ma pewne wady i zalety. Wybór techniki zależy od zbioru danych, problemu i dostępnych zasobów. Oprócz danych niezrównoważonych problemem bywają też dane skorelowane, na przykład podczas pracy z szeregami czasowymi. Przyjrzymy im się w następnym podrozdziale.

Dane skorelowane

Skorelowane dane szeregów czasowych bywają kłopotliwe dla modeli uczenia maszynowego, ponieważ tradycyjne techniki, takie jak próbkowanie losowe, mogą wprowadzać stronniczość i ukrywać zależności między punktami danych. Oto kilka metod, które mogą okazać się pomocne:

- **Walidacja krzyżowa szeregów czasowych**. Dane w szeregach czasowych są często zależne od przeszłych wartości, więc trzeba zachować tę zależność podczas treningu i ewaluacji modelu. Walidacja krzyżowa szeregów czasowych polega na podzieleniu danych na wiele krotności, z których każda stanowi ciągły blok czasu. Podejście to gwarantuje, że model będzie trenowany na przeszłych danych i oceniany na przyszłych, co lepiej symuluje działanie modelu w rzeczywistych scenariuszach.
- **Inżynieria cech**. Skorelowane dane szeregów czasowych trudno modeluje się z użyciem tradycyjnych algorytmów uczenia maszynowego. Inżynieria cech może pomóc przekształcić dane w odpowiedniejszy format. Przykłady inżynierii cech na potrzeby szeregów czasowych to m.in. tworzenie opóźnień lub różnic w szeregu czasowym, agregowanie danych w przedziały i okna oraz tworzenie statystyk ruchomych, takich jak średnia ruchoma.
- **Modele specyficzne dla szeregów czasowych**. Istnieje kilka modeli zaprojektowanych specjalnie z myślą o szeregach czasowych, m.in. **AutoRegressive Integrated Moving Average** (**ARIMA**), **Seasonal ARIMA** (**SARIMA**), **Prophet** oraz sieci **Long Short-Term Memory** (**LSTM**). Modele te potrafią wychwytywać zależności i wzorce w szeregach czasowych i mogą działać lepiej niż tradycyjne modele uczenia maszynowego.
- **Techniki wstępnego przetwarzania szeregów czasowych**. Szeregi czasowe można wstępnie przetworzyć w celu usunięcia korelacji i dostosowania danych do potrzeb modeli uczenia maszynowego. Techniki takie jak różnicowanie, usuwanie trendów i normalizacja mogą pomóc w usunięciu tendencji i składników sezonowych z danych, co może przyczynić się do zmniejszenia korelacji.

- **Techniki zmniejszania wymiarowości**. Skorelowane dane szeregów czasowych często mają wiele wymiarów, co utrudnia modelowanie. Techniki zmniejszania wymiarowości, takie jak PCA lub autoenkodery, pomagają ograniczyć liczbę zmiennych w danych przy zachowaniu najważniejszych informacji.

Ogólnie rzecz biorąc, dane szeregów czasowych należy modelować z użyciem technik, które zachowują temporalne zależności i wzorce w danych. Może to wymagać specjalnych technik modelowania i przetwarzania wstępnego.

Podsumowanie

W tym rozdziale poznałeś różne koncepcje związane z uczeniem maszynowym, zaczynając od eksploracji danych i technik przetwarzania wstępnego. Następnie zbadaliśmy różne modele uczenia maszynowego, takie jak regresja logistyczna, drzewa decyzyjne, maszyny wektorów nośnych i lasy losowe, wraz z ich mocnymi i słabymi stronami. Omówiliśmy również znaczenie podziału danych na zbiory treningowe i testowe, a także techniki obsługi niezrównoważonych zestawów danych.

W rozdziale tym opisaliśmy również koncepcje obciążenia, wariancji oraz niedostatecznego i nadmiernego dopasowania, a także sposoby diagnozowania i rozwiązywania tych problemów. Przyjrzeliśmy się też metodom zespołowym, takim jak bagging, wzmacnianie i spiętrzanie, które mogą poprawić działanie modelu poprzez łączenie prognoz wielu modeli.

Na koniec wspomnieliśmy o ograniczeniach uczenia maszynowego, w tym o potrzebie posiadania dużej ilości danych wysokiej jakości, zagrożeniach związanych ze stronniczością oraz trudnościach w interpretowaniu złożonych modeli. Pomimo tych wyzwań uczenie maszynowe oferuje potężne narzędzia do rozwiązywania szerokiej gamy problemów i wszystko wskazuje na to, że odmieni oblicze wielu branż i dziedzin nauki.

W następnym rozdziale omówimy wstępne przetwarzanie tekstu, które jest wymagane do używania tekstu przez modele uczenia maszynowego.

Bibliografia

- Shahriari, B., Swersky, K., Wang, Z., Adams, R.P., de Freitas, N., *Taking the human out of the loop: A review of Bayesian optimization. Proceedings of the IEEE 104(1)*, s. 148 – 175 (2016). DOI 10.1109/JPROC.2015.2494218.

ODKRYJ PRZYSZŁE TRENDY W NLP WIDZIANE OCZAMI EKSPERTÓW!

Uczenie maszynowe i duże modele językowe rewolucjonizują biznes i nasze codzienne życie. Potencjał tych innowacji jest trudny do oszacowania: modele LLM stały się wiodącym trendem w tworzeniu aplikacji i analizie danych. Integrowanie zaawansowanych modeli z systemami produkcyjnymi bywa jednak często wymagającym, a nawet niewdzięcznym zadaniem.

Na szczęście dzięki tej książce poradzisz sobie z takimi wyzwaniami! Najpierw zapoznasz się z matematycznymi podstawami algorytmów ML i NLP. Zaznajomisz się również z ogólnymi technikami uczenia maszynowego i dowiesz się, w jakim stopniu dotyczą one dużych modeli językowych. Kolejnym zagadnieniem będzie przetwarzanie danych tekstowych, w tym metody przygotowywania tekstu do analizy, po czym przyswoisz zasady klasyfikowania tekstu. Ponadto poznasz zaawansowane aspekty teorii, projektowania i stosowania LLM, wreszcie — przyszłe trendy w NLP. Aby zdobyć praktyczne umiejętności, będziesz ćwiczyć na przykładach rzeczywistych zagadnień biznesowych i rozwiązań NLP.

W książce:

- podstawy matematyczne uczenia maszynowego i NLP
- zaawansowane techniki przetwarzania wstępnego i analizy danych tekstowych
- projektowanie systemów ML i NLP w Pythonie
- przetwarzanie tekstu z użyciem metod uczenia głębokiego
- modele LLM i ich implementacja w różnych aplikacjach AI
- trendy w NLP i potencjał tej technologii

LIOR GAZIT jest ekspertem w zakresie NLP i twórcą wybitnych produktów ML. Jest też szanowanym liderem w branży. Korzysta ze zdobytej wiedzy i bogatego doświadczenia do napędzania pozytywnych zmian w swoich organizacjach.

DR MEYSAM GHAFFARI jest specjalistą data science z dużym doświadczeniem w NLP i uczeniu głębokim. Obecnie tworzy modele ML i NLP na potrzeby opieki zdrowotnej. W przeszłości pracował jako adiunkt na University of Wisconsin-Madison.

Helion
helion.pl
HELION S.A.
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl
KOD KORZYŚCI
Sięgnij po więcej!
ISBN 978-83-289-2048-4
9 788328 920484
Cena: 89,00 zł