**Analiza porównawcza modeli uczenia maszynowego dla klasyfikacji i regresji**

**Wstęp**

Celem tego projektu jest analiza i porównanie skuteczności różnych modeli uczenia maszynowego dla trzech problemów: klasyfikacji binarnej, klasyfikacji wieloklasowej oraz regresji. W analizie wykorzystano trzy algorytmy: k-najbliższych sąsiadów (kNN), drzewa decyzyjne oraz sztuczne sieci neuronowe. Modele te zostały ocenione zarówno w wersji własnoręcznie zaimplementowanej, jak i wbudowanej w popularne pakiety R, takie jak **caret**, **rpart**, **nnet**.

**Zakres analizy**

Analiza obejmuje następujące etapy:

1. **Wybór zbiorów danych** – Dane zostały pobrane z repozytorium Kaggle i podzielone według typów problemów:
   * **Regresja:** Analiza długości życia na podstawie cech demograficznych i społecznych.
   * **Klasyfikacja binarna:** Ocena jakości owoców bananów na podstawie cech sensorycznych.
   * **Klasyfikacja wieloklasowa:** Analiza poziomu zaangazowania w grę.
2. **Przygotowanie danych** – Dane zostały oczyszczone, uzupełnione brakujące wartości oraz dostosowane do wymagań modeli.
3. **Implementacja modeli** – Każdy z trzech problemów został rozwiązany przy użyciu trzech algorytmów:
   * **k-Najbliższych Sąsiadów (kNN)**
   * **Drzew Decyzyjnych**
   * **Sztucznych Sieci Neuronowych** Algorytmy te zostały zaimplementowane ręcznie w języku R oraz porównane z gotowymi implementacjami z pakietów R.
4. **Dostrajanie hiperparametrów** – W celu uzyskania jak najlepszej jakości predykcji, każdy model został poddany strojeniu hiperparametrów, takich jak liczba sąsiadów w kNN, głębokość lub minimalna liczba obserwacji w liściu dla drzewa decyzyjnego czy liczba neuronów w warstwach ukrytych sieci neuronowych.
5. **Ocena i porównanie modeli** – Wyniki modeli zostały ocenione za pomocą różnych miar, takich jak:
   * **Regresja:** MAE, MSE, MAPE
   * **Klasyfikacja:** Accuracy, Precision, Recall, F1-score

Dla każdego ze zbiorów użyłem do analizy 1000 losowo wybranych obserwacji, a dla knn po 200 aby skrocic czas potrzebny do wykonania obliczeń

## ****Regresja – Wstęp****

W analizie regresji wykorzystano zbiór danych **"Life Expectancy Data"**, który zawiera informacje o długości życia w różnych krajach oraz zestaw zmiennych demograficznych, społecznych i ekonomicznych. Celem regresji było przewidzenie wartości długości życia na podstawie dostępnych cech.

### **Wykorzystane techniki i narzędzia:**

1. **Modele regresyjne**
   * k-Najbliższych Sąsiadów (kNN)
   * Drzewa Decyzyjne
   * Sztuczne Sieci Neuronowe
2. **Przygotowanie danych:**
   * Uzupełnienie brakujących wartości medianą dla zmiennych numerycznych
   * Normalizacja predyktorów metodą **z-score**
   * Usunięcie silnie skorelowanych zmiennych
3. **Tuning hiperparametrów:**
   * **kNN:** liczba sąsiadów (k) testowana w zakresie od 1 do 15
   * **Drzewa decyzyjne:** głębokość drzewa i minimalna liczba obserwacji w liściu
   * **Sieci neuronowe:** liczba warstw ukrytych, szybkość uczenia (learning rate), liczba iteracji
4. **Porównanie wyników:**
   * Implementacja własnych modeli porównana z gotowymi modelami z pakietów **caret**, **rpart**, **nnet**
   * Ewaluacja na podstawie miar **RMSE, MAE, MSE, MAPE**

W kolejnych sekcjach omówiono szczegółowe wyniki i wpływ hiperparametrów na jakość predykcji modeli.

**Wyniki:**

W implementacji mojego algorytmu knn użyłem odległości i sprawdziłem hiperparametr k najbliższych sąsiadów.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| k | MAE\_t | RMSE\_t | MAPE\_t | MAE\_w | RMSE\_w | MAPE\_w | MAE\_diff | RMSE\_diff | MAPE\_diff |
| 1 | 0 | 0 | 0 | 3.547 | 5.365528 | 5.250258 | 3.547 | 5.365528 | 5.250258 |
| 3 | 2.056542 | 2.860346 | 3.030963 | 3.016333 | 4.171902 | 4.451145 | 0.9597917 | 1.311556 | 1.420183 |
| 5 | 2.425225 | 3.357147 | 3.576701 | 2.9633 | 4.222921 | 4.390696 | 0.538075 | 0.8657743 | 0.8139958 |
| 7 | 2.560518 | 3.573398 | 3.780808 | 2.969071 | 4.166306 | 4.400983 | 0.4085536 | 0.5929077 | 0.6201753 |
| 9 | 2.700389 | 3.739317 | 4.015417 | 3.023778 | 4.171166 | 4.50129 | 0.3233889 | 0.4318491 | 0.4858729 |
| 11 | 2.803545 | 3.886348 | 4.188349 | 3.071045 | 4.259796 | 4.584059 | 0.2675 | 0.3734474 | 0.3957094 |
| 13 | 2.932923 | 4.04357 | 4.390291 | 3.159538 | 4.32305 | 4.711619 | 0.2266154 | 0.2794803 | 0.3213281 |
| 15 | 3.026683 | 4.191195 | 4.541898 | 3.1914 | 4.379966 | 4.772981 | 0.1647167 | 0.1887711 | 0.231083 |

A zatem można zauważyć, że modele im bardziej zwiększała się liczba sąsiadów tym większą stabilnością cechował się model ( jednak kosztem większych wartości błędów. Tu cięzko jednoznacznie stwierdzić co jest lepsze, zależy od tego jak chcemy podejść do problemu.Jednak jeżeli zależy nam na stabilności To k = 15 zdecydowanie dawało najlepsze wyniki.

**Wyniki pakietu wbudowanego caret:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| k | MAE\_t | RMSE\_t | MAPE\_t | MAE\_w | RMSE\_w | MAPE\_w | MAE\_diff | RMSE\_diff | MAPE\_diff |
| 1 | 0 | 0 | 0 | 2.84092 | 21.03309 | 4.344071 | 2.84092 | 21.03309 | 4.344071 |
| 3 | 1.807124 | 7.433686 | 2.775531 | 2.760377 | 16.77383 | 4.246802 | 0.953253 | 9.340144 | 1.471271 |
| 5 | 2.153529 | 9.967468 | 3.317257 | 2.85644 | 17.12167 | 4.402195 | 0.702911 | 7.154202 | 1.084938 |
| 7 | 2.36912 | 11.61115 | 3.650699 | 2.862871 | 16.95144 | 4.421566 | 0.493751 | 5.340289 | 0.770867 |
| 9 | 2.524303 | 13.02615 | 3.902682 | 2.940102 | 17.55973 | 4.55496 | 0.415799 | 4.533584 | 0.652278 |
| 11 | 2.617234 | 13.73706 | 4.051818 | 2.96669 | 17.55501 | 4.599805 | 0.349456 | 3.817955 | 0.547987 |
| 13 | 2.683558 | 14.32467 | 4.160595 | 2.970547 | 17.54637 | 4.609999 | 0.286989 | 3.221696 | 0.449404 |
| 15 | 2.749482 | 14.95833 | 4.270803 | 2.983824 | 17.56903 | 4.63497 | 0.234342 | 2.6107 | 0.364167 |

Dla pakietu wbudowanego uzyskałem niższe wartości MaE oraz Mape za to RMSE było wyższe. Dla k 15 również otrzymujemy najlepszą stabilność, jednakże ku mojemu zdziwieniu gorszą niż w mojej funkcji. Być może wynika to z małej liczby obserwacji ograniczonej do 200.

Drzewa decyzyjne:

W przypadku moich drzew jako hiperparametry użyłem maksymalną głebokość oraz liczbe minimalnych obserwacji jako hiperparametry.

Wyniki:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| max\_depth | minobs | MAE\_t | RMSE\_t | MAPE\_t | MAE\_w | RMSE\_w | MAPE\_w | MAE\_diff | RMSE\_diff | MAPE\_diff |
| 1 | 5 | 5.096372 | 6.834739 | 7.92629 | 5.168029 | 6.904465 | 8.018183 | 0.071657 | 0.0697258 | 0.0918933 |
| 2 | 5 | 4.103254 | 5.504691 | 6.331912 | 4.291953 | 5.705483 | 6.605919 | 0.188699 | 0.2007918 | 0.274007 |
| 3 | 5 | 3.060425 | 4.243406 | 4.721926 | 3.335725 | 4.585434 | 5.14342 | 0.2752995 | 0.3420273 | 0.4214935 |
| 4 | 5 | 2.300784 | 3.30858 | 3.496935 | 2.613842 | 3.743049 | 3.99631 | 0.3130579 | 0.4344692 | 0.4993754 |
| 5 | 5 | 1.883368 | 2.630863 | 2.824022 | 2.387138 | 3.532029 | 3.645473 | 0.5037701 | 0.9011662 | 0.8214509 |

Można zauważyć, że tu wraz ze wzrostem głębokości spada stabilność ale i maleją błedy. W takim wypadku dla maksymalnej glebokości = 1 i minobs = 5 otrzymujemy najlepszą stabilność modelu.

Metoda wbudowana Rpart:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| minobs | depth | MAE\_t | RMSE\_t | MAPE\_t | MAE\_w | RMSE\_w | MAPE\_w | MAE\_diff | RMSE\_diff | MAPE\_diff |
| 1 | 5 | 5.008882 | 45.442127 | 7.736352 | 5.198322 | 48.62537 | 8.04777 | 0.1894407 | 3.183243 | 0.311418 |
| 2 | 5 | 3.937721 | 27.270174 | 5.955659 | 4.080419 | 29.25068 | 6.189525 | 0.1426981 | 1.980506 | 0.233866 |
| 3 | 5 | 2.862044 | 15.371145 | 4.368918 | 3.028744 | 17.11366 | 4.631216 | 0.1667004 | 1.742515 | 0.262298 |
| 4 | 5 | 2.356574 | 10.872046 | 3.542004 | 2.78835 | 15.44999 | 4.243345 | 0.4317767 | 4.577944 | 0.701341 |
| 5 | 5 | 2.036667 | 8.008587 | 3.01495 | 2.620311 | 13.9124 | 3.995896 | 0.5836437 | 5.903813 | 0.980946 |

W tym wypadku największą stabilność modelu mamy dla minobs 1 i glebokości 5. Są to gorsze wartości niż te dla naszego modelu.

Sieci neuronowe:

W sieciach iterowalem po liczbie neuronow i learning rate ze stala liczba iteracji i seedem. Oto wyniki:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| h | lr | iter | seed | MAE\_t | RMSE\_t | MAPE\_t |
| 5,5 | 0.01 | 5000 | 123 | 8.206806 | 10.36962 | 12.59092 |
| 10,10 | 0.01 | 5000 | 123 | 7.606233 | 9.780592 | 11.76569 |
| 15,15 | 0.01 | 5000 | 123 | 7.75602 | 9.622663 | 11.89588 |
| 5,5 | 0.005 | 5000 | 123 | 8.500699 | 10.76926 | 12.85669 |
| 10,10 | 0 | 5000 | 123 | 9.652597 | 12.11309 | 14.92234 |
| 15,15 | 0 | 5000 | 123 | 9.89311 | 12.1173 | 15.02958 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| MAE\_w | RMSE\_w | MAPE\_w | MAE\_diff | RMSE\_diff | MAPE\_diff |
| 8.32537 | 10.48287 | 12.78649 | 0.118564 | 0.113247 | 0.195562 |
| 7.924031 | 10.09518 | 12.266 | 0.317798 | 0.314591 | 0.500308 |
| 7.924986 | 9.857443 | 12.16605 | 0.168965 | 0.23478 | 0.270176 |
| 8.510803 | 10.8239 | 12.85856 | 0.010103 | 0.054641 | 0.001874 |
| 9.776737 | 12.30434 | 15.08819 | 0.12414 | 0.191243 | 0.165854 |
| 9.935164 | 12.20674 | 15.06946 | 0.042054 | 0.089439 | 0.039873 |

Tutaj najbardziej stabilny model wyszedł mi dla learning rate 0.005 i kombinacji neuronow 5,5.

**Sieci neuronowe pakiet rpart:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| size | decay | MAE | MSE | MAPE |
| 5 | 0.01 | 1.79819 | 6.132493 | 2.689375 |
| 5 | 0.005 | 1.860272 | 6.50872 | 2.784101 |
| 10 | 0.01 | 1.641198 | 5.372872 | 2.489261 |
| 10 | 0.005 | 1.665736 | 5.489081 | 2.528336 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| MAE\_val | MSE\_val | MAPE\_val | MAE\_diff | MSE\_diff | MAPE\_diff |
| 2.575724 | 14.15708 | 3.895051 | 0.777534 | 8.024587 | 1.205676 |
| 2.71366 | 16.65254 | 4.06568 | 0.853388 | 10.14382 | 1.281579 |
| 3.223105 | 25.22396 | 4.871526 | 1.581907 | 19.85109 | 2.382265 |
| 3.393238 | 27.48039 | 5.224611 | 1.727502 | 21.99131 | 2.696275 |

Tutaj najlepsze wyniki wychodzily dla size 5 i decay 0,01,. A przynajmniej bardziej stabilne. Niestety nie można tutaj miarodajnie prowownac która metoda była lepsza bo maja inne hiperparametry.

### **Analiza wyników i porównanie modeli uczenia maszynowego**

Klasyfikacja binarna – Banana Quality

Dla problemu klasyfikacji binarnej, który polegał na ocenie jakości owoców bananów, zastosowano algorytmy k-najbliższych sąsiadów (kNN), drzew decyzyjnych oraz sieci neuronowych.

**kNN**  
W przypadku modelu własnego najlepsze wyniki uzyskano dla **k=11**, gdzie dokładność wyniosła **96,12%**, a wskaźnik F1 **0.96**. Porównując to z implementacją wbudowaną, najlepsze wyniki uzyskano dla **k=15**, z dokładnością **96,76%**. Ostatecznie, modele własne i wbudowane wykazały podobne skuteczności, jednak model wbudowany okazał się nieco stabilniejszy.

**Drzewa decyzyjne**  
W przypadku drzew decyzyjnych najlepsze wyniki osiągnięto dla **max\_depth = 1 i minobs = 5**, uzyskując dokładność **69,57%** i F1-score **0.60**. Dla modelu wbudowanego uzyskano gorsze wyniki – **66,79% dokładności** i **0.55 F1-score**. Widać, że model własny lepiej dopasowywał się do danych.

**Sieci neuronowe**  
Sieci neuronowe wykazały pewne problemy w optymalizacji. Najlepszy wynik uzyskano dla architektury **(5,5) i learning rate = 0.01**, gdzie dokładność wyniosła **95,5%**, a F1-score **0.95**. Model wbudowany osiągnął bardzo zbliżone wyniki, co sugeruje, że obie wersje były równie skuteczne.

Klasyfikacja Wieloklasowa

W analizie klasyfikacji wieloklasowej badano poziom zaangażowania graczy w gry. Modele oceniano na podstawie metryk: Accuracy, Macro F1 i Recall.

**kNN**  
Najlepsze wyniki uzyskano dla **k=15**, gdzie dokładność wyniosła **67,1%**, a Macro F1 **0.67**. Model wbudowany uzyskał zbliżone wyniki, jednak przy mniejszych wartościach k zauważalne były duże różnice między zbiorami treningowymi i testowymi.

**Drzewa decyzyjne**  
Dla własnego modelu najlepsze wyniki uzyskano przy **max\_depth=1**, co dało dokładność **57,1%** i Macro F1 **0.66**. W przypadku modelu wbudowanego najlepszy wynik uzyskano przy **max\_depth=5**, gdzie dokładność wyniosła **82,7%** i Macro F1 **0.83**, co wskazuje, że model wbudowany radził sobie lepiej z danymi wieloklasowymi.

**Sieci neuronowe**  
Najlepsze wyniki uzyskano dla modelu **(15,15) i learning rate = 0.005**, gdzie dokładność wyniosła **47,5%**, a Macro F1 **0.53**. Model wbudowany uzyskał lepszą dokładność **79,8%**, ale różnice w Macro F1 nie były aż tak duże, co wskazuje na lepszą generalizację modelu wbudowanego.

**Podsumowanie:**

* W **klasyfikacji binarnej** najlepsze wyniki uzyskano dla kNN i sieci neuronowych, gdzie modele wbudowane były minimalnie lepsze.
* W **klasyfikacji wieloklasowej** drzewa decyzyjne i sieci neuronowe wbudowane uzyskały lepsze wyniki niż modele własne.
* W **regresji** modele wbudowane miały lepsze wartości błędów, szczególnie dla sieci neuronowych, gdzie różnice były znaczące.

Podsumowując, modele własne w wielu przypadkach uzyskiwały porównywalne wyniki do modeli wbudowanych, jednak w klasyfikacji wieloklasowej i regresji modele wbudowane wykazywały lepszą generalizację i stabilność.

Paweł Kubiak s208522