Informatyka w medycynie – siatkówka oka

1. **Język programowania:**
   -python
   biblioteki:

- matplotlib
- cv2
- tabulate
- numpy
- sklearn
- imblearn
- sklearnex
- skiimage

Link do repozytorium kodu: https://github.com/Prorokslawek/eye-blood-vessel

2. Zastosowane metody:

Przetwarzanie obrazów

Wczytanie obrazu i konwersja przestrzeni barw

- Obraz jest wczytywany za pomocą cv2.imread i konwertowany z BGR do RGB
- Wyświetlane są poszczególne kanały (R, G, B) oraz obraz w skali szarości
- Wykorzystanie kanału zielonego (G) jako podstawy dalszej analizy, ponieważ naczynia krwionośne są najbardziej widoczne w tym kanale

Zastosowanie CLAHE – wyrównanie histogramu (Contrast Limited Adaptive Histogram Equalization),który ma na celu poprawę kontrastu obrazu

- Wybór kanału zielonego i ograniczenie wartości pikseli do zakresu 10-245
- Zastosowanie CLAHE z parametrami clipLimit=2.0(próg ograniczenia kontrastu) i tileGridSize=(8, 8) (8×8=64 kafelki)
- CLAHE poprawia lokalny kontrast obrazu, co jest kluczowe dla uwydatnienia naczyń krwionośnych przy zachowaniu szczegółów

Zastosowanie filtru Frangiego

- Filtr Frangiego jest stosowany z parametrem sigmas=np.arange(1, 5, 0.5)(od 1 do 5 z korkiem co pół) i black_ridges=True(wykrywa ciemne grzbiety na białym tle)
- Normalizacja wyniku i progowanie z wartością 0.065(img_frangi > 0.065, 1, 0)
- Usunięcie obramowania obrazu za pomocą funkcji remove_border
- Filtr Frangiego jest specjalnie zaprojektowany do wykrywania struktur rurowych (jak naczynia krwionośne) poprzez analizę wartości własnych macierzy Hessego

Sekwencyjne operacje morfologiczne otwarcia i zamknięcia

- Zastosowanie elementów strukturalnych w kształcie elipsy o różnych rozmiarach (5, 7, 15, 23)
- Operacje otwarcia usuwają małe obiekty i szumy
- Operacje zamknięcia wypełniają małe dziury i łączą bliskie struktury

- Sekwencyjne zwiększanie rozmiaru jądra pozwala na stopniowe usuwanie coraz większych struktur

Usuwanie tła

Usunięcie tła z obrazu

- Wykorzystanie ostatniego obrazu z sekwencji operacji morfologicznych jako przybliżenia tła
- Odjęcie tła od oryginalnego obrazu w skali szarości za pomocą cv2.subtract
- Zastosowanie rozmycia Gaussa (5x5) do wygładzenia wyniku
- Aby uczynić naczynia jeszcze bardziej widocznymi, zdecydowaliśmy się zastosować rozmycie Gaussa w celu usunięcia szumów z obrazu oryginalnego

Normalizacja obrazu po usunięciu tła

- Ograniczenie wartości pikseli do zakresu 0-20
- Normalizacja do pełnego zakresu 0-255
- Zastosowanie CLAHE dla poprawy kontrastu do obrazu po usunięciu tła
- Normalizacja poprawia kontrast i ułatwia segmentację naczyń

Binaryzacja i porównanie z maską eksperta

- Dla celów porównania obrazów, przyjmujemy, że (w skali od 0 do 1), piksele o wartości powyżej 0,75 otrzymują wartość 1, natomiast piksele o wartościach poniżej tej granicy są traktowane jako 0. Jest to binarna maska odpowiedzi algorytmu.

Ewaluacja wyników

Obliczenie metryk oceny jakości segmentacji

- Obliczenie macierzy pomyłek
- Wyliczenie metryk:
    1. dokładność (accuracy) oznacza, że X% wszystkich pikseli zostało poprawnie sklasyfikowanych
    2. czułość (sensitivity) oznacza, że X% pikseli naczyń zostało poprawnie wykrytych
    3. swoistość (specificity) oznacza, że X% pikseli tła zostało poprawnie zidentyfikowanych jako tło
    4. zbalansowana dokładność, to średnia arytmetyczna czułości i swoistości
    5. średnia geometryczna, to pierwiastek kwadratowy z iloczynu czułości i swoistości
- Metryki te są kluczowe do obiektywnej oceny jakości segmentacji, szczególnie w przypadku niezbalansowanych danych (gdzie piksele naczyń stanowią mniejszość)

Uczenie maszynowe

W procesie przygotowania danych zastosowano metodę wycinków (patch-based approach), gdzie z każdego obrazu siatkówki pobierano fragmenty o wymiarach 5×5 pikseli. Dla każdego wycinka przeprowadzono ekstrakcję następujących cech:

1. Podstawowe statystyki:
    - Wariancja pikseli w wycinku
    - Średnia wartość pikseli
    - Odchylenie standardowe
2. Wartości ekstremalne:
    - Minimalna wartość piksela
    - Maksymalna wartość piksela
    - Zakres (różnica między wartością maksymalną a minimalną)
3. Momenty obrazu:
    - Momenty Hu (7 wartości)
    - Momenty centralne (mu20, mu11, mu02, mu30, mu21, mu12, mu03)

1. mu20, mu02: Opisują rozproszenie (wariancję) obiektu wzdłuż osi x i y. Związane są z orientacją obiektu.
2. mu11: Moment mieszany, który wskazuje na korelację między wartościami x i y, dostarczając informacji o orientacji obiektu.
3. mu30, mu03: Momenty trzeciego rzędu, które opisują skośność (asymetrię) wzdłuż osi x i y.
4. mu21, mu12: Mieszane momenty trzeciego rzędu, które dostarczają dodatkowych informacji o złożoności kształtu obiektu.

Ekstrakcja cech została zaimplementowana w funkcji extract_features(), która przekształca każdy wycinek 5×5 w wektor cech o stałej długości. Etykiety dla każdego wycinka określano na podstawie wartości centralnego piksela w masce eksperta (wartość >128 klasyfikowana jako naczynie krwionośne).

Wstępne przetwarzanie zbioru uczącego

Przed przystąpieniem do uczenia modelu, przeprowadzono następujące kroki przetwarzania wstępnego:

1. Przetwarzanie obrazów wejściowych:
    - Wykorzystanie kanału zielonego jako najbardziej informatywnego dla struktur naczyniowych

- Ograniczenie wartości pikseli do zakresu 10-245
- Zastosowanie rozmycia Gaussa (5×5) w celu redukcji szumów
- Normalizacja kontrastu za pomocą CLAHE

2. Balansowanie klas:
    - Zastosowanie techniki **Random Undersampling** – w celu zrównoważenia liczby próbek naczyń i tła

      rus = RandomUnderSampler(sampling_strategy=1, random_state=42) oznacza, że stosunek będzie wynosił 1:1
    - Redukcja zbioru uczącego do maksymalnie 30000 próbek w celu optymalizacji czasu uczenia

3. Podział danych:
    - Wykorzystano 10 pierwszych obrazów jako zbiór treningowy
    - Zastosowano podział train_test_split z parametrem test_size=0.3 do wydzielenia zbioru walidacyjnego

W projekcie zastosowano klasyfikator **Random Forest** z optymalizacją hiperparametrów za pomocą przeszukiwania siatki (Grid Search). Przeszukiwanie przeprowadzono dla następujących parametrów:

- n_estimators: - liczba drzew w lesie
- max_depth:- maksymalna głębokość drzewa
- min_samples_leaf: - minimalna liczba próbek wymagana w liściu
- min_samples_split: - minimalna liczba próbek wymagana do podziału węzła

  Optymalizacja została przeprowadzona z wykorzystaniem **3-krotnej walidacji krzyżowej** (cv=3):

1. W każdej iteracji, 2 z 3 części danych są używane do trenowania modelu
2. Pozostała 1 część służy jako zbiór walidacyjny do oceny wydajności modelu
3. Proces jest powtarzany 3 razy, tak aby każda z 3 części danych raz posłużyła jako zbiór walidacyjny

Cały proces uczenia został zaimplementowany w postaci potoku (pipeline) z wykorzystaniem biblioteki imblearn, co pozwoliło na zintegrowanie procesu balansowania klas z procesem uczenia. Wyniki wychodzą podobne przy zastosowaniu scalera StandardScaler(), dlatego zdecydowaliśmy się na pominięcie tego kroku.

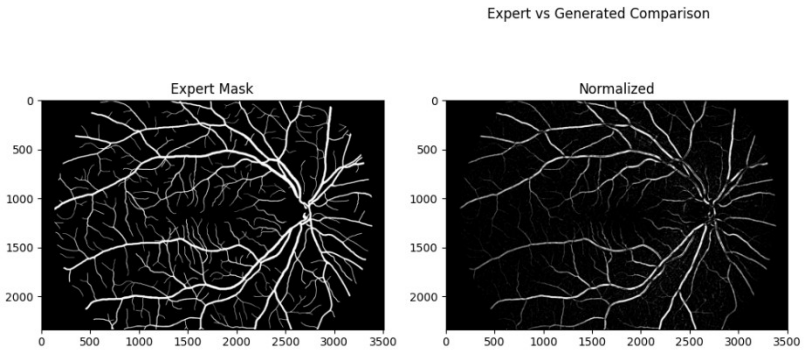
Aby przeprowadzić proces dostrajania modelu, tworzymy obiekt GridSearchCV, który będzie przeszukiwać siatkę parametrów w celu znalezienia najlepszego zestawu dla modelu. Parametr n_jobs=-1 pozwala na wykorzystanie wszystkich dostępnych rdzeni procesora, przyspieszając tym samym proces dostrajania.

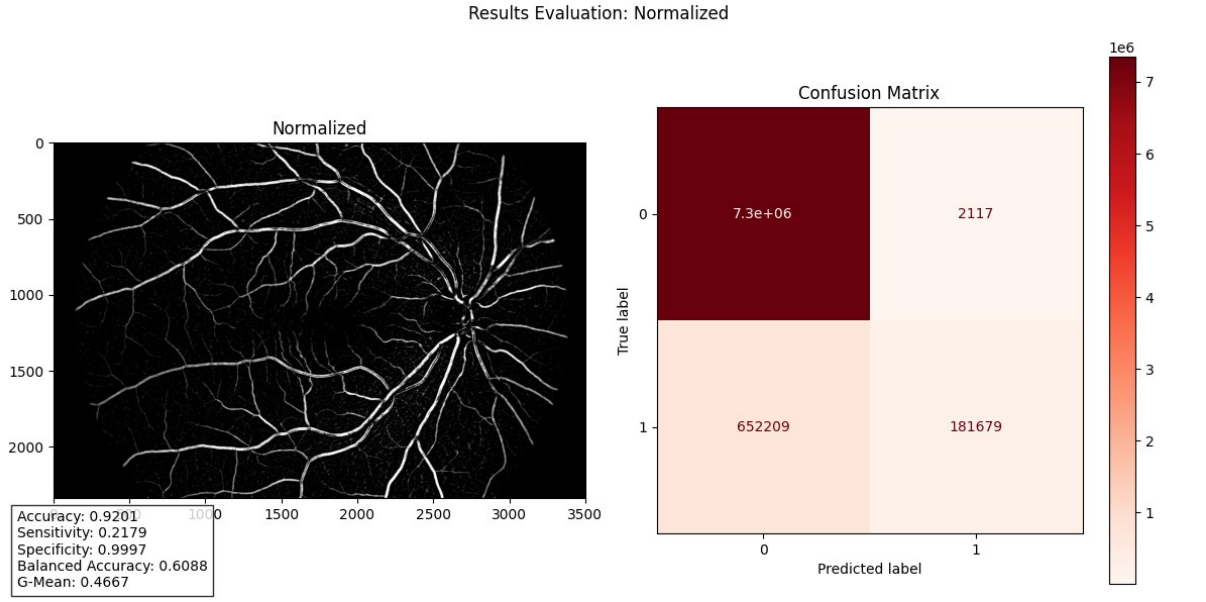
Zastosowane rozwiązanie opiera się na podejściu **patch-based z klasyfikatorem Random Forest**, co jest uzasadnione z następujących powodów:

1. Efektywność obliczeniowa - Random Forest oferuje dobrą wydajność przy relatywnie niskich kosztach obliczeniowych w porównaniu do głębokich sieci neuronowych
2. Odporność na przeuczenie - Random Forest jest mniej podatny na przeuczenie dzięki agregacji wielu drzew decyzyjnych, co jest istotne przy ograniczonej liczbie obrazów treningowych
3. Skuteczność przy danych niezbalansowanych - Po zastosowaniu technik balansowania, Random Forest dobrze radzi sobie z problemem niezbalansowanych klas, co jest typowe dla segmentacji naczyń siatkówki (gdzie piksele naczyń stanowią mniejszość).
4. Zastosowanie kanału zielonego oraz technik przetwarzania wstępnego (CLAHE, rozmycie Gaussa) jest zgodne z najlepszymi praktykami w przetwarzaniu obrazów siatkówki, gdzie naczynia krwionośne są najlepiej widoczne właśnie w tym kanale.

Wyniki testów hold-out potwierdzają skuteczność zastosowanego podejścia, szczególnie w kontekście wysokiej swoistości, co jest istotne w zastosowaniach medycznych, gdzie fałszywe pozytywne wyniki mogą prowadzić do błędnych diagnoz. Analizując uzyskane wyniki, można stwierdzić, że istnieje potencjał do dalszej poprawy. Jednym z czynników, który mógłby wpłynąć na lepsze rezultaty, jest zwiększenie liczby próbek tła. Chociaż RandomUnderSampler pozwala na zrównoważenie klas, jego parametry ograniczają się do zakresu 0-1, co może wpłynąć na ogólną jakość modelu.

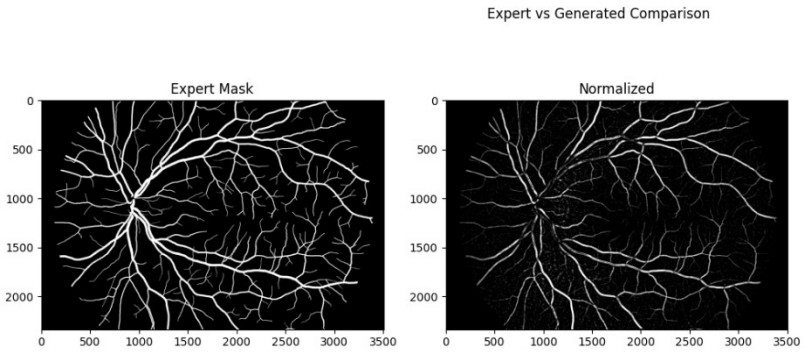
Obraz 1:

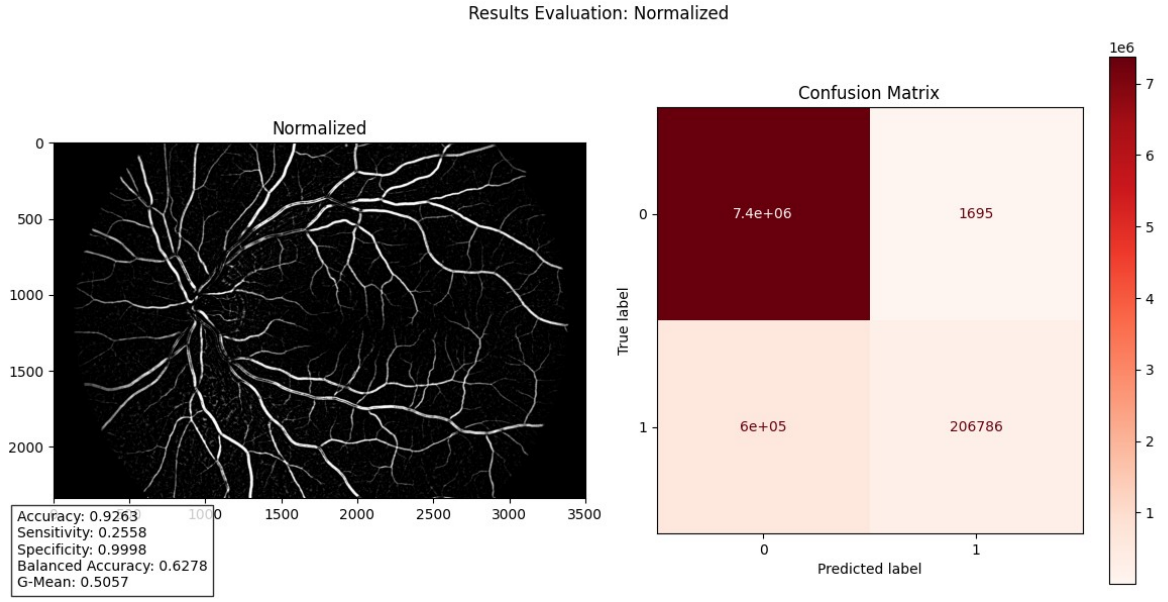




Method	Accuracy	Sensitivity	Specificity	Balanced Accuracy	G-Mean
Frangi	0.9048	0.1082	0.9951	0.5517	0.3282
Normalized	0.9201	0.2179	0.9997	0.6088	0.4667

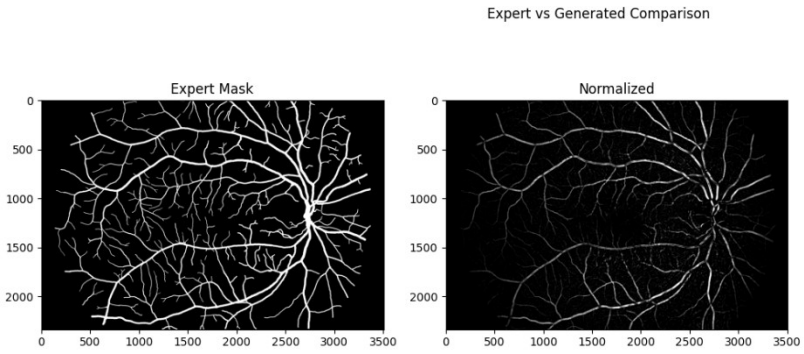
Obraz 2:

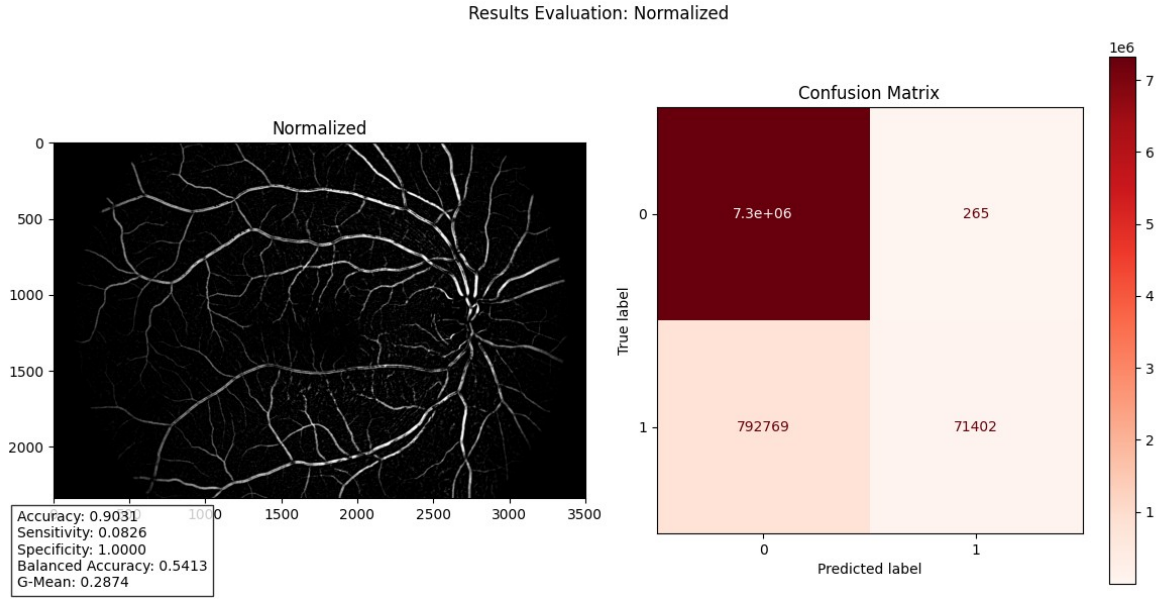





```
C:\python3\python.exe "C:\Users\Sławek\Desktop\studia\sem_6\Informatyka_w_medycynie\siatkowka oka\main.py"
| Method     |   Accuracy |   Sensitivity |   Specificity |   Balanced Accuracy |   G-Mean |
|:-----------|-----------:|--------------:|--------------:|--------------------:|---------:|
| Frangi     |     0.9052 |        0.0576 |        0.9981 |              0.5279 |   0.2397 |
| Normalized |     0.9263 |        0.2558 |        0.9998 |              0.6278 |   0.5057 |
```

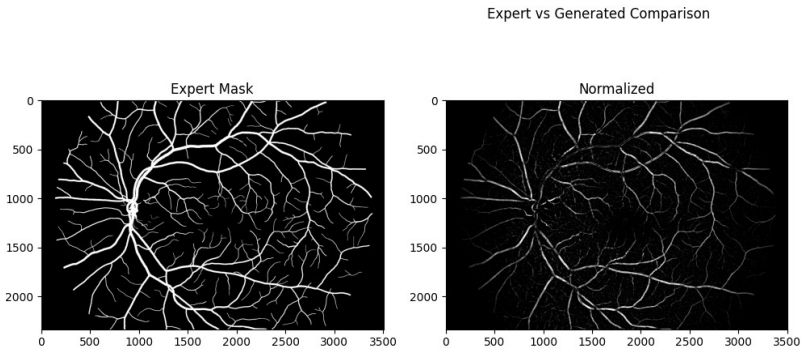
Obraz 3:

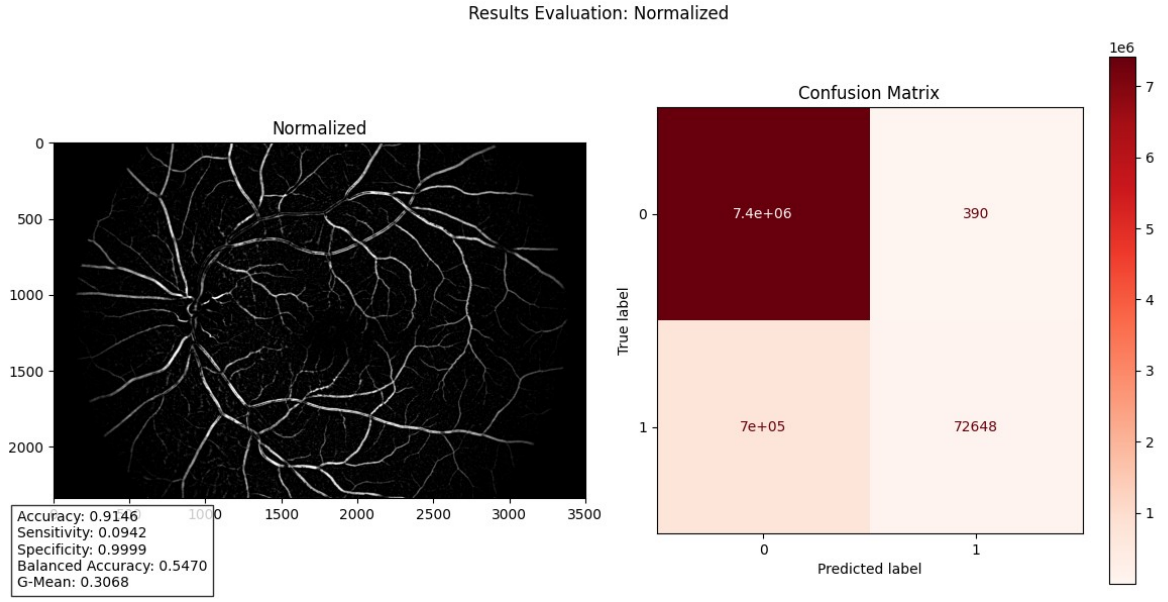




Method	Accuracy	Sensitivity	Specificity	Balanced Accuracy	G-Mean
Frangi	0.8967	0.0732	0.9939	0.5335	0.2697
Normalized	0.9031	0.0826	1	0.5413	0.2874

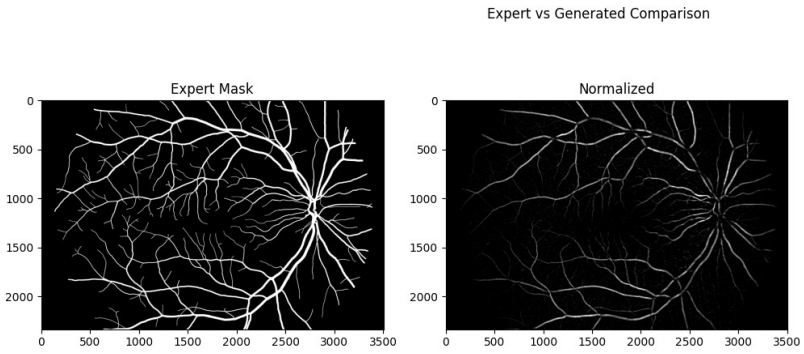
Obraz 4:

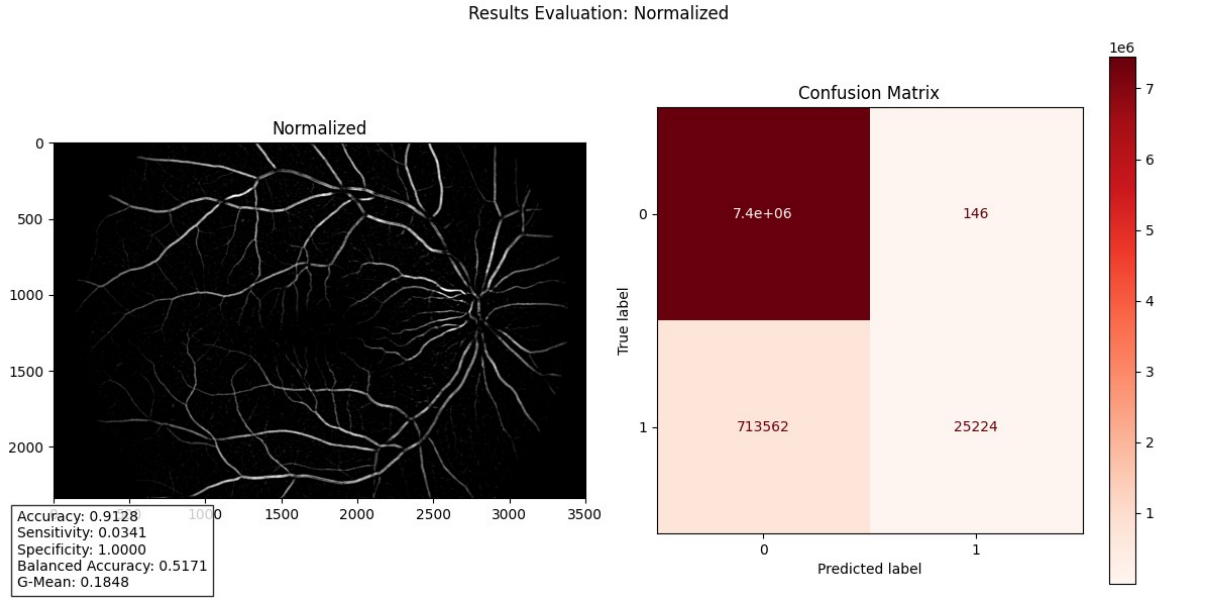




Method	Accuracy	Sensitivity	Specificity	Balanced Accuracy	G-Mean
Frangi	0.9094	0.0862	0.9951	0.5406	0.2929
Normalized	0.9146	0.0942	0.9999	0.547	0.3068

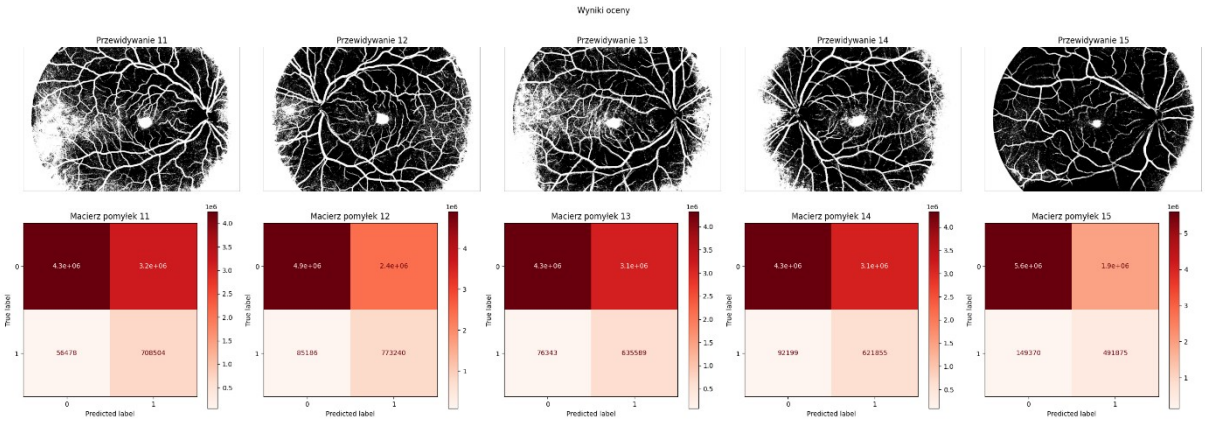
Obraz 5:

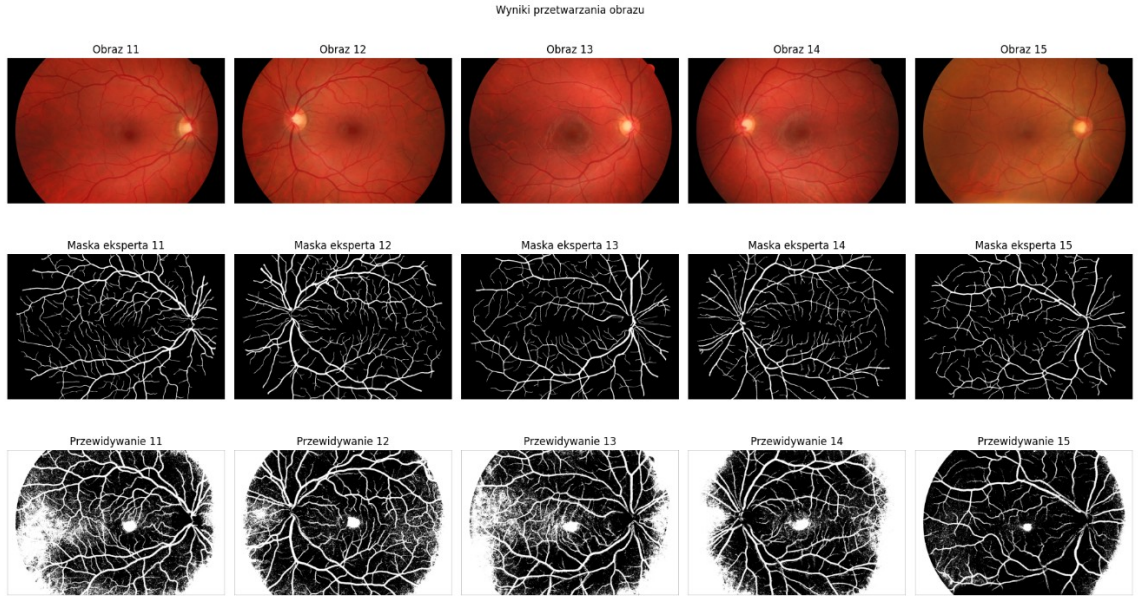




Method	Accuracy	Sensitivity	Specificity	Balanced Accuracy	G-Mean
Frangi	0.9087	0.0415	0.9947	0.5181	0.2031
Normalized	0.9128	0.0341	1	0.5171	0.1848

Uczenie maszynowe:






```
Trenowanie nowego modelu...
Rozmiar zbioru treningowego:  29997800
Kształt X_train:  (29997800, 20)
Kształt y_train:  (29997800,)
y_train etykieta 1 (naczynia):  3722164
y_train etykieta 0 (tło):  26275636
Rozmiar zbioru treningowego po próbkowaniu:  7444328
y_train etykieta 1 (naczynia):  3722164
y_train etykieta 0 (tło):  3722164
Rozmiar zbioru treningowego po redukcji:  30000
y_train etykieta 1 (naczynia):  15102
y_train etykieta 0 (tło):  14898
Najlepsze parametry:  {'rfc__max_depth': 20, 'rfc__min_samples_leaf': 1, 'rfc__min_samples_split': 5, 'rfc__n_estimators': 100}
Najlepszy wynik:  0.852
Dokładność treningu: 0.9731333333333333
Dokładność walidacji: 0.8741015230005756
Model zapisany do vessel_segmentation_model.joblib

Metryki oceny zbioru testowego:
| Metoda          |   Dokładność |   Czułość |   Swoistość |   Zrównoważona dokładność |   G-średnia |
|:----------------|-------------:|----------:|------------:|--------------------------:|------------:|
| Obraz 11        |       0.6066 |    0.9262 |      0.5737 |                    0.7499 |      0.7289 |
| Obraz 12        |       0.6934 |    0.9008 |      0.6691 |                    0.7849 |      0.7763 |
| Obraz 13        |       0.6066 |    0.8928 |      0.5793 |                    0.736  |      0.7192 |
| Obraz 14        |       0.6061 |    0.8709 |      0.5808 |                    0.7259 |      0.7112 |
| Obraz 15        |       0.7451 |    0.7671 |      0.7432 |                    0.7551 |      0.755  |
| Średnia testowa |       0.6516 |    0.8716 |      0.6292 |                    0.7504 |      0.7381 |

Process finished with exit code 0
```