**. Dokumentacja usług Web Services**

### API do pobierania zdjęć z zewnętrznych serwerów

Poprzez publiczne API dostępna jest funkcja, która pobiera zdjęcie z zewnętrznego serwera i zwraca je zakodowane w formacie base64.

**Przykład użycia - wywołanie** Metoda HTTP POST.

**Parametry wywołanie:** url - adres URL zdjęcia, które chcemy pobrać

**Przykładowe wywołanie (za pomoc biblioteki jQuery):**

|  |
| --- |
| $.ajax({  ’type’:’POST’,  ’contentType’: ’application/json; charset=utf-8’,  ’dataType’: ’json’,  ’data’: {’url’: [adres URL zdjecia]},  ’success’: function(data) {success\_callback(data);},  ’error’: function(data) { error\_callback(data); },  ’url’: ’http://[adres serwera]/api/get\_image’  }); |

1

2

**Przykładowa odpowiedz (skrócona):**

|  |
| --- |
| {"img":"data:image/jpeg;base64,/9j/4AAQSkZJRgABAQEASABIAAD/2 wBDAAMCAgMCAgMDAwMEAwMEBQgFBQQEBQoHBwYIDAoMDAsKCwsNDhIQDQ4RDgsLEBYQERMUFRUVDA8 XGBYUGBIUFRT ...  /nBBCFwoZSLpPW8LXDsj+0ZukSxtaafSyb/AMSwj9YIIcamwvoBlpRuRaal2 xlbYSGkDwSkZQPkBGQTeCCHJtwnivUqtHrnSCCI64qgnLFCRcmCCBCrIuLR5kHnBBAhf//Z"} |

## 4 Testy jednostkowe

Moduł *face\_utils* to interfejs zawierający funkcje wykorzystywane przez kilka klas w obrębie wytwarzanej aplikacji. Praca nad tym modułem przebiegała zgodnie z metodologią *Test Driven Development*, tj. przed powstaniem danej jego części (funkcji) powstawał zestaw testów, według którego opracowywany był kod.

Praca nad dana funkcjonalnością rozpoczynała się od określenia przypadku testowego (odzwierciedlany jest on nazwa metody testującej). Metody testujące początkowo powstały bez implementacji. Następnie po utworzeniu pustych funkcjonalności powstała implementacja testów, których uruchomienie zgodnie z oczekiwaniami zakończyło się niepowodzeniem. Następnie metoda przyrostowa tworzone były pojedyncze minimalne funkcjonalności, które były poddawane testowaniu i refaktoryzacji. W ten sposób w duchu metodologii TDD powstały dobrze przetestowane funkcjonalności, zgodne ze specyfikacja dostarczona przez klienta.

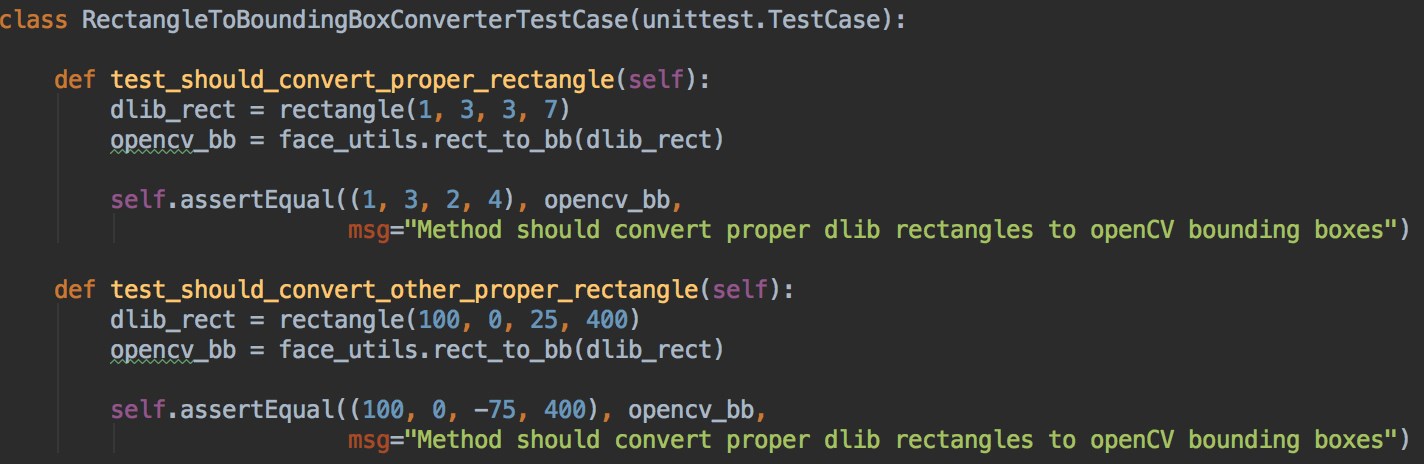
Moduł face\_utils składa się z trzech funkcji:

* rect\_to\_bb(rect) - przekształcenie prostokąta zgodnego z biblioteka dlib do bounding boxa zgodnego z OpenCV
* load\_csv\_data(path) - wczytanie współrzędnych punktów oraz etykiet klas (identyfikatorów osób) z katalogu danego parametrem path
* filter\_sets(x, y) - odfiltrowanie ze zbiorów x, y etykiet (identyfikatorów osób) i danych dla etykiet, które maja mniej niż trzy wektory w zbiorze x.

Testy jednostkowe opracowane zostały przy użyciu najbardziej popularnej biblioteki *unittest*, stosowanej w większości projektów w języku Python. Udostępnia ona szeroki wybór sprawdzeń (asserts), które pasują do rozważanych przypadków testowych.

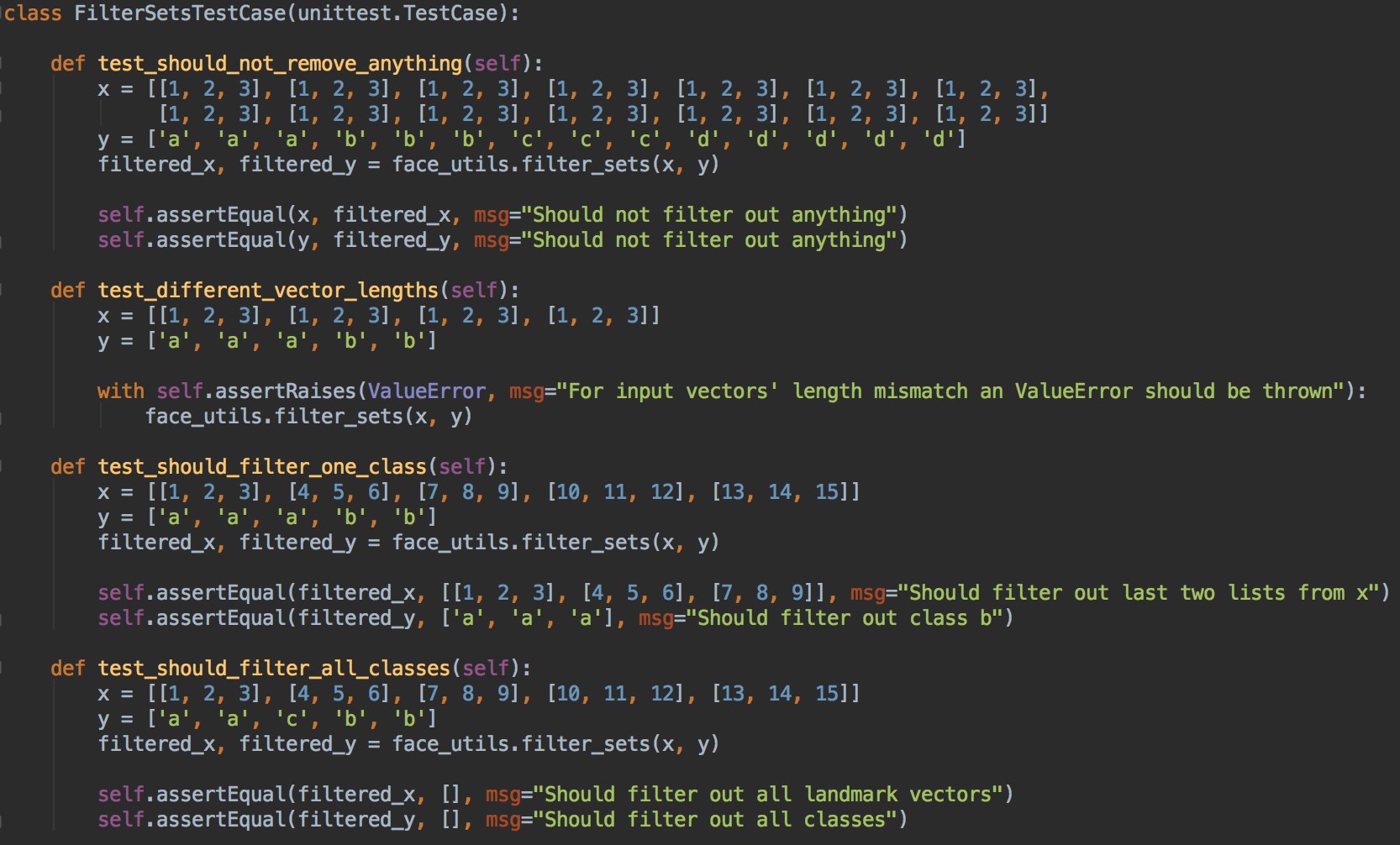
W module *face\_utils\_tests* znajdują się trzy klasy, każda z nich zawiera zestaw testów dla odpowiedniej części modułu *face\_utils*.

• RectangleToBoundingBoxConverterTestCase - testy poprawności konwersji dla prostokątów, sprawdzenie poprawności algorytmu.



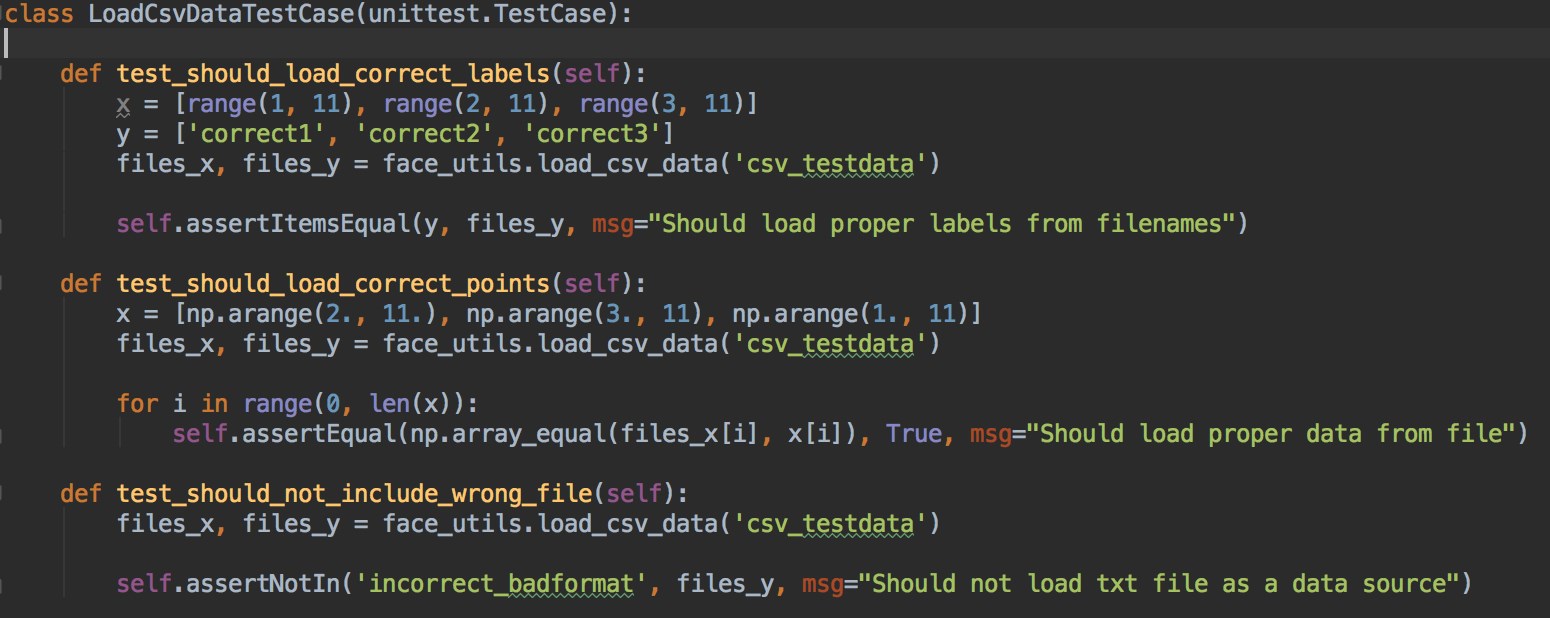
### Rysunek 20: Testy konwersji

• FilterSetsTestCase - testy poprawności filtrowania zbiorów. Przypadki testowe to brak usunięcia żadnej osoby, sprawdzenie poprawności danych wejściowych (czy funkcja rzuca wyjątek dla niepoprawnych danych), odfiltrowywanie dokładnie jednej etykiety i wszystkich etykiet.



### Rysunek 21: Testy filtrowania danych

• LoadCsvDataTestCase - testy poprawności ładowania danych z plików csv z podziałem na ładowanie etykiet, ładowanie danych, a także sprawdzenie, czy funkcja pomija pliki w błędnymi formacie.



Rysunek 22: Testy ładowania danych z plików.

## 5 Testy integracyjne

Aplikacja budowana jest w sposób modularny, tzn. wydzielonych zostało kilka komponentów, które mogły być implementowane niezależnie przez kilku programistów. W związku z tym, po zakończeniu pracy nad danym modułem obowiązkiem jego autora było przeprowadzenie manualnych lub automatycznych testów integracyjnych.

Tworzenie aplikacji przebiegało metoda bottom-up, tzn. najpierw powstały bazowe moduły. Dla modułów implementujących interfejs Recognizer, tj. AnnRecognizer, SvmRecognizer, GpRecognizer, wymagana była pełna integracja z fasadaRequestProcessor. Kolejnym elementem tworzonej aplikacji był interfejs użytkownika, który musiał zostać przetestowany pod względem integracji z API udostępnianym przez serwer.

Osobnym modułem jest interfejs administracyjny, dla którego wymagane było sprawdzenie poprawności połączenia zarówno z implementacjami interfejsu Recognizer (ponieważ wymaganiem jest, aby administrator systemu mógł trenować, importowa c i eksportowa c klasyfikatory), a także metodami pomocniczymi do operacji na zbiorach danych wejściowych.

## 6 Plan testów akceptacyjnych

Testy akceptacyjne stanowi ostatni etap testowania naszej aplikacji. Pozwalają uzyskać odpowiedz na to, czy aplikacja działa według określonych wcześniej przez zespół wytycznych. Testujemy tutaj nie tyle poszczególne komponenty aplikacji czy poprawność ich współdziałania w systemie, ile faktyczne funkcjonalności, które posiada nasza aplikacja.

W związku z tym, testy akceptacyjne zostały przygotowane w formie scenariuszy testowych, które mogą zostać wykonane przez programistę, testera lub osobę nadzorującą wykonywanie projektu. Dają one tym samym odpowiedz pozytywna/negatywna niemalże natychmiastowo - jest to przydatne zwłaszcza w przypadku dodawania większych zmian, które istotnie mogą wpłynąć na działanie programu.

### 6.1 Poziom usprawnień: użytkownik

|  |  |  |
| --- | --- | --- |
| **Nazwa testu** | **Opis** | **Rezultat testu** |
| Strona główna powinna się poprawnie wyświetlać | Nawigacja do strony głównej aplikacji. | Po wpisaniu adresu aplikacji w przeglądarce Firefox (w wersji 56 lub nowszej) lub Chrome (w wersji 61 lub nowszej) wyświetla się strona nagłówkiem: Praca inżynierska: *Aplikacja do identyfikacji osób na podstawie zdjęć udostępnionych w sieci* i *logiem* wydziału MiNI PW. |
| Aplikacja pozwala na upload zdjęcia z dysku użytkownika | Jedna z metod uploadu pliku. | Po pewnym czasie wyżarzony obszar z napisem *NO PHOTO SELECTED* zawiera przeskalowane zdjęcie wybrane przez użytkownika. Pod obszarem *drag & drop* pojawia się przycisk *Remove*. |
| Aplikacja pozwala na upload zdjęcia metoda drag & drop | Jedna z metod uploadu pliku. | Po pewnym czasie wyżarzony obszar z napisem *NO PHOTO SELECTED* zawiera przeskalowane zdjęcie wybrane przez użytkownika. Pod obszarem *drag & drop* pojawia się przycisk *Remove*. |
| Aplikacja pozwala na upload zdjęcia poprzez podanie URL | Jedna z metod uploadu pliku. | Po pewnym czasie wyżarzony obszar z napisem *NO PHOTO SELECTED* zawiera przeskalowane zdjęcie wybrane przez użytkownika. Pod obszarem *drag & drop* pojawia się przycisk *Remove*. |
| Aplikacja pozwala na usuniecie wrzuconego przez użytkownika zdjęcia. | Metoda cofnięcia wprowadzonego do systemu zdjęcia. | Zdjęcie wrzucone przez użytkownika znika i pojawia się wyżarzony obszar z napisem *NO PHOTO SELECTED*. |
| Aplikacja pozwala na zlecenie identyfikacji metoda ANN, SVM, GP lub wszystkimi trzema jednocześnie. | Zlecenie klasyfikacji wybrana metoda. | Po uploadzie zdjęcia i wyborze metody identyfikacji przycisk *Identify* staje się aktywny. |
| Aplikacja poprawnie obsługuje zdjęcia nie zawierające twarzy. | Użytkownik informowany jest o niepoprawnym zdjęciu. | Przekierowanie do strony wynikowej z napisem *No faces detected* i dostępnym przyciskiem *Try again*. |
| Aplikacja poprawnie obsługuje zdjęcia zawierające jedna twarz. | Poprawne przetwarzanie fotografii – z dokładnie jedną twarzą | Przekierowanie do strony wynikowej. Strona wynikowa zawiera zdjęcie z prostokątem ograniczającym obszar twarzy i kropkami oznaczającymi punkty charakterystyczne twarzy, a także tabele z kolumnami *Method*, *Identifier*, *Probability*, *Time*. |
| Aplikacja poprawnie obsługuje zdjęcia zawierające wiele twarzy. | Poprawne przetwarzanie fotografii – z więcej niż jedną twarzą | Przekierowanie do strony wynikowej. Strona wynikowa zawiera zdjęcie z prostokątami ograniczającymi obszary twarzy i kropkami oznaczającymi punkty charakterystyczne twarzy, a także tabele z kolumnami *Method*, *Identifier*, *Probability*, *Time* dla każdej znalezionej twarzy. |
| Aplikacja poprawnie rozpoznaje twarze ze zbioru treningowego. | Poprawne przetwarzanie fotografii ze zbioru treningowego. | Przekierowanie do strony wynikowej. Identyfikator osoby jest zgodny z oczekiwanym (tj. równy nazwie zdjęcia). Prawdopodobieństwo przekracza próg klasy- fikacji ustalony przez administratora. |

Tablica 7: Testy akceptacyjne dla części aplikacji dostępnej dla użytkownika

### 6.2 Poziom usprawnień: administrator

|  |  |  |
| --- | --- | --- |
| **Nazwa testu** | **Opis** | **Rezultat testu** |
| Aplikacja powinna poprawnie podzielić dane na zbiory: treningowy, testowy i walidacyjny, zgodnie z proporcjami w pliku settings.json. | Poprawny  preprocessing danych wejściowych. | Utworzenie katalogów wskazanych w polach training\_data\_path,  testing\_data\_path,  validation\_data\_path, wewnątrz których znajdują się zdjęcia we wskazanych proporcjach. |
| Aplikacja powinna poprawnie generować pliki csv na podstawie zdjęć w zbiorach: treningowym, testowym i walidacyjnym | Poprawny  preprocessing  danych wejściowych. | Utworzenie katalogów csv\_46 i csv\_68 wewnątrz katalogów wskazanych w polach training\_data\_path, testing\_data\_path oraz validation\_data\_path. Wewnątrz tych katalogów powinny znajdować się pliki csv z współrzędnymi punktów charakterystycznych twarzy. |
| Aplikacja powinna importować klasy- fikatory zgodnie z ustawieniami w pliku settings.json. | Poprawne wczytywanie wygenerowanych klasyfikatorów. | Wybrany klasyfikator powinien natychmiast zostać wczytany z pliku, bez oczekiwania na wytrenowanie nowego klasyfikatora. |
| Aplikacja powinna eksportować klasyfikatory zgodnie z ustawieniami w pliku settings.json. | Poprawne generowanie  wytrenowanego klasyfikatora | Pojawienie się pliku z wytrenowanym klasyfikatorem w podanej ścieżce. |
| Aplikacja powinna trenować klasyfikatory, jeśli nie wybrano opcji importowania klasyfikatora. | Poprawne przygotowanie klasyfikatorów  przed uruchomieniem serwera. | Wytrenowane klasyfikatory pozwalają na identyfikacje osób po uruchomieniu serwera. |

Tablica 8: Testy akceptacyjne dla części aplikacji dostępnej dla administratora

## 7 Instrukcja obsługi

**7.1 Instrukcja administratora**

### Plik konfiguracyjny

Aplikacja wykorzystuje ustawienia wczytywane z pliku JSON w systemie plików użytkownika. W korzeniu pliku konfiguracyjnego znajdują się trzy sekcje: **ann**, **svm**, **gp** a także kilka ustawień dotyczących zbioru danych wejściowych:

|  |  |
| --- | --- |
| {  } | "training\_data\_path": "data/training",  "testing\_data\_path": "data/testing",  "validation\_data\_path": "data/validation",  "face\_files": ["faces94", "faces95"],  "should\_generate\_datasets": false,  "should\_generate\_csv": false,  "training\_size": 0.7,  "testing\_size": 0.15,  "validation\_size": 0.15,  "ann": {}  "svm": {}  "gp": {} |

1. **training\_data\_path**, **testing\_data\_path**, **validation\_data\_path** - określają ścieżki do katalogów zawierających zbiory treningowy, testowy oraz walidacyjny.
2. **face\_files** - tablica zawierająca napisy - nazwy archiwów (bez rozszerzeń) w formacie zip, zawierających dane wejściowe.
3. **should\_generate\_datasets** - flaga boolowska określającą, czy przed uruchomieniem serwera aplikacji powinny zostać wygenerowane zbiory treningowy, testowy i walidacyjny na podstawie archiwów podanych jako **face\_files**. Jeżeli na serwerze nie wygenerowano uprzednio takich zbiorów, to wymagane jest ustawienie tej flagi na true.
4. **should\_generate\_csv** - aplikacja pracuje na plikach csv, a nie zdjęciach w formatach graficznych. W związku z tym konieczne jest przekonwertowanie zdjęć na format csv. Jeżeli na serwerze nie wygenerowano uprzednio takich plików, to wymagane jest ustawienie tej flagi na true.
5. **training\_size**, **testing\_size**, **validation\_size** - w przypadku generowania zbiorów od zera, są to proporcje miedzy ich licznościami, określane jako ułamek wszystkich dostępnych zdjęć. Zalecane jest, by liczność zbioru treningowego była większa od połowy wszystkich zdjęć.

Każda z poniższych sekcji pliku konfiguracyjnego zawiera dwie tożsame podsekcje: pierwsza dotyczy pracy z rozpoznawaniem na podstawie 68, druga - 46 punktów charakterystycznych. Omówiona zostanie tylko jedna z nich, jako że zawierają one dokładnie te same pola.

### Sekcja ann

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| "ann": | {  "68": | {  "classify\_threshold": | 0.85, | |  |  |
|  |  | "activation": "relu", "hidden\_layer\_sizes": | [100, 200, 300], | |  |  |
|  |  | "alpha": 0.001,  "max\_iter": 300,  "should\_use\_grid\_search": false,  "should\_import\_classifier": true,  "should\_export\_classifier": false,  "import\_file\_path": "ann\_classifier\_68.pkl",  "export\_file\_path": "ann\_classifier\_68.pkl", "grid\_search\_parameter\_grid": {  "alpha": [0.01, 0.001, 0.0001, 1e-4, 1e-5, 1e-7], | | |  |  |
|  |  | "activation": ["relu"],  "hidden\_layer\_sizes": [[7, 14,  80], [100, 200, 300]],  "max\_iter": [100, 200, 300]  } | | 21], [20, 30, 40], | [40, 60, |  |
| } | }, "46": {} | | | |  |  |

1. **classify\_threshold** - progowa wartość prawdopodobienstwa. Jeżeli się zwróci etykietę osoby z prawdopodobieństwem *p* mniejszym niż zadany próg, to odpowiedziasystemu na zapytanie użytkownika jest para etykieta “unclassified” i prawdopodobienstwo 1 − *p*.
2. **activation** - funkcja aktywacji uzywana przez siec neuronowa. Do wyboru saponizsze funkcje:
   1. ‘identity’, *f*(*x*) = *x*
   2. ‘logistic’, *f*(*x*) = 1+1*e*−*x*.
   3. ‘tanh’, *f*(*x*) = *tanh*(*x*).
   4. ‘relu’, *f*(*x*) = *max*(0*,x*)
3. **hidden\_layer\_sizes** - tablica z liczba neuronów w kolejnych warstwach ukrytych sieci. Trzyelementowa tablica [*x,y,z*] oznacza, że sieć ma trzy warstwy ukryte, z których pierwsza ma *x* neuronów, druga *y*, a trzecia *z*.
4. **alpha** - kara w rozumieniu normy *L*2
5. **max\_iter** - siec jest trenowana aż do uzyskania zbieznosci lub do tej liczby iteracji.
6. **should\_use\_grid\_search** - flaga boolowska, okresla czy wykorzystywany jest algorytm grid search, który wyszukuje najlepszakombinacje sposród parametrów danych w obiekcie **grid\_search\_parameter\_grid**
7. **should\_import\_classifier**, **should\_export\_classifier** - flagi boolowskie okreslajace czy klasy- fikator powinien zostać zaimportowany/wyeksportowany przed uruchomieniem aplikacji.
8. **import\_file\_path**, **export\_file\_path** - ścieżki do importu/eksportu klasyfikatora.
9. **grid\_search\_parameter\_grid** - obiekt zawierający dla kazdego pola tablice dozwolonych wartosci, z których podczas treningu zostanie dobrana najlepsza kombinacja, o ile parametr **should\_use\_grid\_search** ma wartosc true.

### Sekcja svm

|  |  |  |  |
| --- | --- | --- | --- |
| "svm": | {  "68": | {  "kernel": "rbf", "c": 32,  "gamma": 0.0039,  "threads\_num": 1,  "classify\_threshold": 0.6,  "should\_import\_classifier": true,  "should\_export\_classifier": false,  "import\_file\_path": "svm\_classifier\_68.pkl",  "export\_file\_path": "svm\_classifier\_68.pkl",  "should\_use\_grid\_search": false, "grid\_search\_parameter\_grid": {  "estimator\_\_kernel": ["rbf"],  "estimator\_\_C": [0.03125, 0.125, 0.5, 2, 8, 32, 128, 512, 2 048, 8192, 32768],  "estimator\_\_gamma": [0.0000305, 0.000061, 0.00012, 0.00024,  0.00049, 0.00098, 0.002, 0.0039, 0.008]  } |  |
| } | },  "46": | {} |  |

1. **kernel** - kernel używany przez algorytm. Wartosc musi by c napisem ze zbioru

### [‘*linear*0*,*‘*poly*0*,*‘*rbf*0*,*‘*sigmoid*0]

1. **c** - parametr wpływajacy na rozmiar marginesu
2. **gamma** - współczynnik kernela dla kerneli [0*rbf*0*,*0 *poly*0*,*0 *sigmoid*0]
3. **threads\_num** - liczba watków uzywanych w procesie treningu.
4. **classify\_threshold** - progowa wartosc prawdopodobnie nstwa. Jeżeli się c zwróci etykiete osoby z prawdopodobieństwem *p* mniejszym niż zadany próg, to odpowiedziasystemu na zapytanie użytkownika jest para etykieta “unclassified” i prawdopodobienstwo 1 − *p*.
5. **should\_use\_grid\_search** - flaga boolowska, okresla czy wykorzystywany jest algorytm grid search, który wyszukuje najlepszakombinacje sposród parametrów danych w obiekcie **grid\_search\_parameter\_grid**
6. **should\_import\_classifier**, **should\_export\_classifier** - flagi boolowskie okreslajace czy klasy- fikator powinien zostać zaimportowany/wyeksportowany przed uruchomieniem aplikacji.
7. **import\_file\_path**, **export\_file\_path** - ścieżki do importu/eksportu klasyfikatora.
8. **grid\_search\_parameter\_grid** - obiekt zawierający dla kazdego pola tablice dozwolonych wartosci, z których podczas treningu zostanie dobrana najlepsza kombinacja, o ile parametr **should\_use\_grid\_search** ma wartosc true.

W rozdziale Model danych zamieszczono dodatkowe wymagania dotyczące zbioru treningowego, o których należy pamiętać, zmieniając ustawienia aplikacji (na przykład przy zmienianiu proporcji podziału zdjęć na zbiory treningowy, testowy i walidacyjny).

#### **Sekcja gp**

|  |  |  |  |
| --- | --- | --- | --- |
| "gp": { "46": | {  "gp\_params": {  "population\_size": 100,  "elite\_size": 5,  "generations": 500,  "evolution\_strategy": "roulette",  "tournament\_size": 5,  "stopping\_criteria": 1.0,  "const\_range": [-2.5, 2.5],  "depth": [2, 25],  "init\_depth": [6, 8],  "init\_method": "full",  "metric": "accuracy",  "function\_set": ["add", "sub", "min", "div", "sqrt", "min", "max", "if"],  "parsimony\_coefficient": 0,  "p\_crossover": 0.8,  "p\_subtree\_mutation": 0.2,  "p\_hoist\_mutation": 0,  "p\_point\_mutation": 0.0,  "p\_point\_replace": 0.0,  "n\_jobs": -1,  "verbose": 1  },  "should\_import\_classifier": true,  "import\_file\_path": "gp\_classifier\_46.pkl", "should\_export\_classifier": false,  "export\_file\_path": "gp\_classifier\_46.pkl",  "should\_use\_grid\_search": false, "grid\_search\_parameter\_grid": {  "evolution\_strategy": ["roulette", "rank", "tournament"],  "tournament\_size": [5, 10, 15], | |  |
|  | "const\_range": [[-1.0, | 1.0], [-2.5, 2.5], [-5.0, 5.0]], |  |
|  | "init\_depth": [[6, 8], | [7, 9], [8, 10]], |  |
|  | "function\_set": [  ["add", "sub", | "mul", "div"], |  |
|  | ["add", "sub", if"] | "mul", "div", "neg", "min", "max", " |  |

|  |  |  |
| --- | --- | --- |
|  |  | ],  "parsimony\_coefficient": [0, 0.000001, 0.00001, 0.0001] |
| } | }  },  "68": {} |  |

1. **gp\_params** - obiekt zawierający parametry algorytmu programowania genetycznego
   1. **population\_size** - rozmiar populacji programów w jednej generacji (bedacy liczbacałkow-

ita)

* 1. **elite\_size** - ilosc najlepszych programów, które przejdado kolejnej populacji
  2. **generations** - maksymalna liczba generacji, którawykona programowanie genetyczne
  3. **evolution\_strategy** - strategia wyboru programów do ewolucji - wartosc ze zbioru [‘*roulette*0*,*‘*rank*0*,*‘*tournament*0]
  4. **tournament\_size** - rozmiar tabeli turniejowej dla strategii turniejowej
  5. **stopping\_criteria** - wartosc funkcji dopasowania, która spowoduje wcze sniejsze zako ncze- nie treningu
  6. **const\_range** - przedział wartosci, z których losowane bedaterminale stałe
  7. **depth** - przedział wyznaczajacy dopuszczalnagłebokosc programu w populacji
  8. **init\_depth** - przedział wyznaczajacy dopuszczalna głebokosc programu generowanego do populacji poczatkowego
  9. **init\_method** - metoda generacji populacji poczatkowej - wartosc ze zbioru



* 1. **metric** - funkcja dopasowania - wartosc ze zbioru 
  2. **function\_set** - zbiór funkcji, na podstawie których generowane saprogramy do populacji

- lista elementów ze zbioru *,*



* 1. **parsimony\_coefficient** - współczynnik kary, któraobarcza się dopasowanie programu w zależności od jego długo sci
  2. **p\_crossover** - prawdopodobienstwo wystapienia operacji crossover
  3. **p\_subtree\_mutation** - prawdopodobienstwo wystapienia operacji mutacji poddrzewa
  4. **p\_hoist\_mutation** - prawdopodobienstwo wystapienia operacji dzwigni
  5. **p\_point\_mutation** - prawdopodobienstwo wystapienia operacji mutacji punktu
  6. **p\_point\_replace** - prawdopodobienstwo wystapienia operacji zamiany punktu
  7. **n\_jobs** - ilosc watków, na które zostanie podzielona ewolucja populacji ( −1 oznacza maksymalnamozliwawartosc)
  8. **verbose** - flaga ustawiajaca tryb verbose (0 lub 1)

1. **should\_use\_grid\_search** - flaga boolowska, okresla czy wykorzystywany jest algorytm grid search, który wyszukuje najlepszakombinacje sposród parametrów danych w obiekcie **grid\_search\_parameter\_grid**
2. **should\_import\_classifier**, **should\_export\_classifier** - flagi boolowskie okreslajace czy klasy- fikator powinien zostać zaimportowany/wyeksportowany przed uruchomieniem aplikacji.
3. **import\_file\_path**, **export\_file\_path** - sciezki do importu/exportu klasyfikatora.
4. **grid\_search\_parameter\_grid** - obiekt zawierający dla kazdego pola tablice dozwolonych wartosci, z których podczas treningu zostanie dobrana najlepsza kombinacja, o ile parametr **should\_use\_grid\_search** ma wartosc true.

##### 7.7.2 Instalacja zależności i uruchomienie serwera aplikacyjnego

Należy upewnić się, że na systemie operacyjnym serwera zainstalowane są następujące biblioteki:

* python-dev
* python-matplotlib
* libboost-all-dev
* build-essential
* cmake
* libfreetype6-dev
* libpng12-dev

W przypadku braku którejkolwiek z powyzszych nalezy zainstalowac ją za pomocą dostepnego menadzera pakietów, np. w Ubuntu:

|  |
| --- |
| $ sudo apt-get install nazwa\_pakietu |

1

Aby zainstalować wymagane przez aplikacje biblioteki (z wyjątkiem tych wymienionych powyżej), należy w katalogu aplikacji wykonać polecenie:

|  |
| --- |
| # ./install.sh |

1

Skrypt ten wykorzystuje mechanizm wirtualny, dzięki czemu całe środowisko aplikacji jest wyizolowane i nie wpływa na biblioteki uprzednio zainstalowane na serwerze aplikacyjnym. Uruchomienie aplikacji polega na wykonaniu poniższego polecenia:

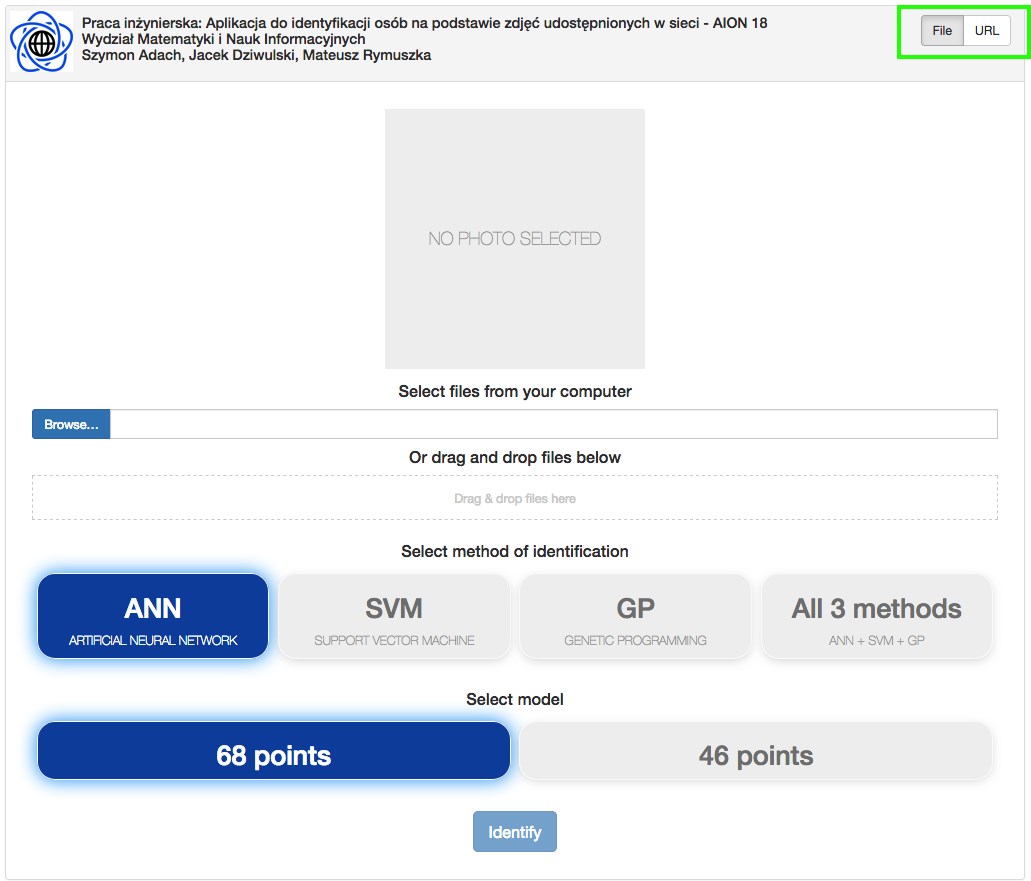
|  |
| --- |
| # ./run.sh path-to-the-settings-file |

1

Parametrem jest do pliku JSON z ustawieniami serwera.

**7.3 Instrukcja użytkownika**

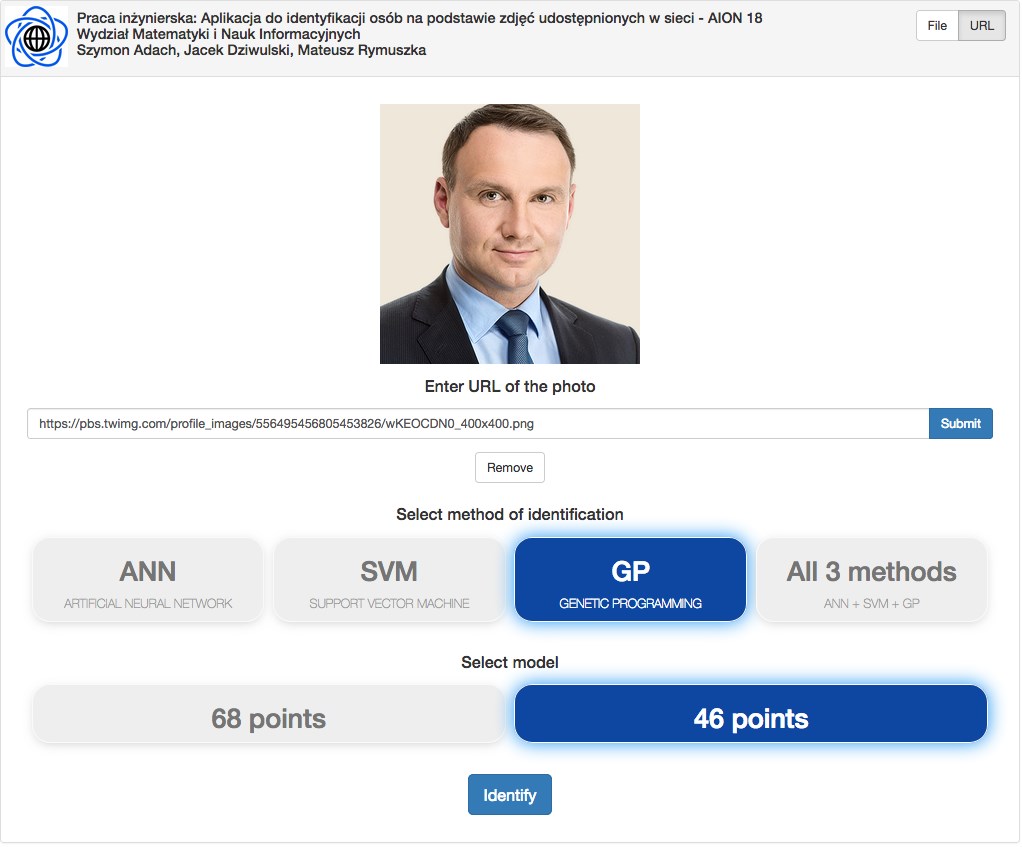
#### **Strona główna aplikacji**



Rysunek 23: Interfejs użytkownika - strona główna aplikacji. Przycisk **’Identify’** jest nieaktywny, ponieważ nie załadowano zdjęcia.

* Przyciski **File/URL** (obszar zaznaczony zielona ramka na rysunku 1)- możliwość wyboru up- loadu pliku z systemu plików użytkownika albo poprzez podanie hiperłącza (linku) do zdjęcia w Internecie.
* Sekcja **’Select files from your computer’** - możliwość wyboru zdjęć z systemu plików użytkownika - po wciśnięciu przycisku **’Browse’** pojawia się standardowe okienko wyboru pliku.
* Sekcja **’Or drag and drop files below’** alternatywna metoda wyboru zdjęcia z systemu plików użytkownika - w dowolnym eksploratorze plików należy wybrać plik graficzny, przeciągnąć go i upuści c w wyznaczonym miejscu (obszar **’Drag & drop files here’**).
* Sekcja **’Select method of identification’** - możliwość wyboru jednej z trzech metod identy- fikacji (sztuczne sieci neuronowe, metoda wektorów wspierających, programowanie genetyczne) lub zlecenia identyfikacji wszystkimi trzema metodami.
* Sekcja ’**Select model’** - wybór jednego z dwóch modeli do wykrywania twarzy: wykorzystującego 68- lub 46 punktów charakterystycznych.
* Przycisk **’Identify’** - jego wciśniecie powoduje zlecenie identyfikacji serwerowi dla danego zdjęcia, przy użyciu wybranej metody i modelu. Przycisk pozostaje nieaktywny, dopóki użytkownik nie załaduje zdjęcia (sygnalizowane jest to zmiana wyżarzonego kwadratu z napisem **’No photo selected’** na wybrane przez użytkownika zdjęcie).

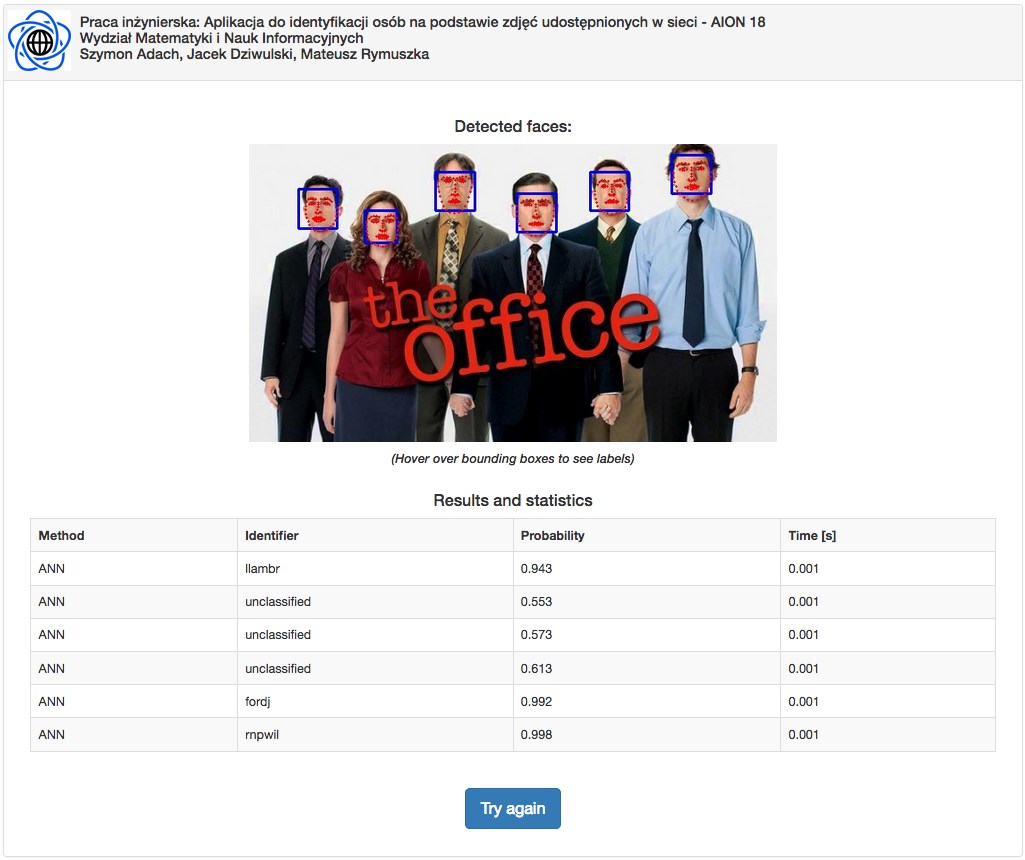
Po wybraniu metody **’URL’** uzytkownikowi przedstawiane jest poniższe menu:



Rysunek 24: Interfejs użytkownika - strona główna aplikacji. Przycisk **’Identify’** jest aktywny, ponieważ poprawnie załadowano zdjęcie z hiperłącza użytkownika. Pojawia się także przycisk **’Remove’**, pozwalający usunąć wrzucone zdjęcie i dokona c nowego uploadu.

W polu tekstowym przy przycisku **’Submit’** należy wprowadzić prawidłowy link prowadzący do zdjęcia. Wcisniecie przycisku powoduje próbę załadowania zdjęcia przez aplikacje. W przypadku błędnego adresu, na dole ekranu pojawia się komunikat tekstowy informujący o tym fakcie. W przypadku, gdy ładowanie zdjęcia nie powiedzie, się, w obszarze prezentacji zdjęcia pojawi się ikona symbolizująca niepoprawny plik graficzny:  .

#### **Strona wynikowa aplikacji**



Rysunek 25: Interfejs użytkownika - strona z wynikami.

Po wciśnięciu przycisku **’Identify’** użytkownik przenoszony jest na stronę wynikowa. Zawiera ona zdjęcie wprowadzone przez użytkownika z naniesionymi markerami dla każdej znalezionej twarzy (niebieski prostka ograniczający twarz oraz czerwone punkty charakterystyczne), oraz tabele zawierająca kolumny: nazwa metody, identyfikator (lub “unclassified”, jeżeli nie udało się rozpoznać osoby - prawdopodobieństwo było zbyt niskie) prawdopodobnie czas potrzebny na rozpoznanie twarzy (lub stwierdzenie że nie da się jej rozpoznać).

Użytkownik może najechać kursorem na wybrany wiersz tabeli, co powoduje jego podświetlenie, a także wyróżnienie markera na zdjęciu. Najechanie kursorem na marker na zdjęciu powoduje wyróżnienie odpowiednich wierszy w tabeli. Możliwe jest także klikniecie w wiersz, wówczas taki wybór jest zapamiętywany i na zdjęciu zaznaczony jest odpowiedni marker.

Użytkownik może powrócić do strony głównej za pomoc przycisku **’Try again’** na dole strony.

## 7.8 Rejestr zmian

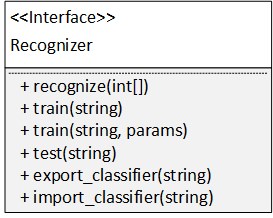
### Zmiana interfejsu Recognizer

**Zmiana:**

Modyfikacja parametrów przyjmowanych przez metody oraz zwracanych obiektów.

**Uzasadnienie:**

W trakcie implementacji okazało się, że do poprawnego działania niezbędne są małe zmiany. Wynik metody test nie będzie zapisywany do pliku, gdyż nie ma takiej potrzeby, będzie on jedynie zwracany w postaci liczby, oznaczającej poprawność przewidywań.



Rysunek 26: Uaktualniona wersja interfejsu Recognizer.

# 8 Porównanie wyników

## 8.1 Wyniki sztucznych sieci neuronowych

Zgodnie z wymaganiami stawianymi przed aplikacja, metodę sztucznych sieci neuronowych administrator systemu może skonfigurować do własnych potrzeb poprzez zmianę proporcji miedzy zbiorami lub zmianę parametrów samej sieci. Możliwe do edycji parametry opisane zostały w instrukcji administratora.

Poniższe testy przeprowadzone zostały na sieci neuronowej w modelu wielowarstwowego perceptronu, wykorzystującej klasę MLPClassifier z biblioteki scikit-learn. Optymalne parametry zostały znalezione za pomocą klasy GridSearchCV z tej samej biblioteki. Na wejściu otrzymuje ona zbiór dozwolonych parametrów, wśród których następnie sprawdzana jest każda kombinacja i zwracana najlepsza. Znalezione dla sieci neuronowej (trenowanej na zbiorze treningowym stanowiącym 70% sumy faces94 i faces95) wartości przedstawia lista poniżej:

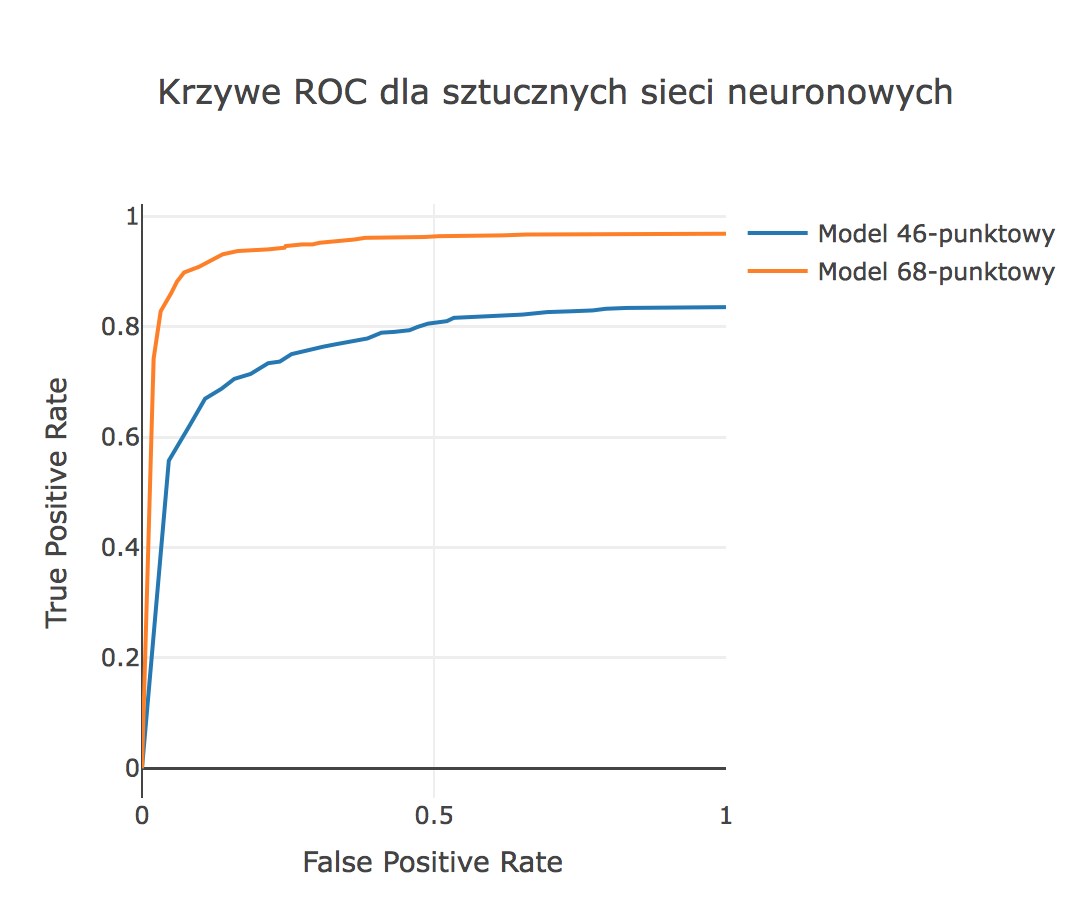
* Liczba warstw i neuronów w warstwach: 3 warstwy liczące kolejno 100, 200, 300 neuronów
* Funkcja aktywacji: *tanh*(*x*)
* Maksymalna liczba iteracji procesu uczenia: 300

### • *α* : 0*.*0001

• Algorytm optymalizacji wag: adam (Adaptive Moment Estimation [23])

Ponadto wartość progu klasyfikacji ustalono na 0*.*90. Poniżej tej wartości prawdopodobieństwa klasyfikacji zwracana jest etykieta „unclassified”.

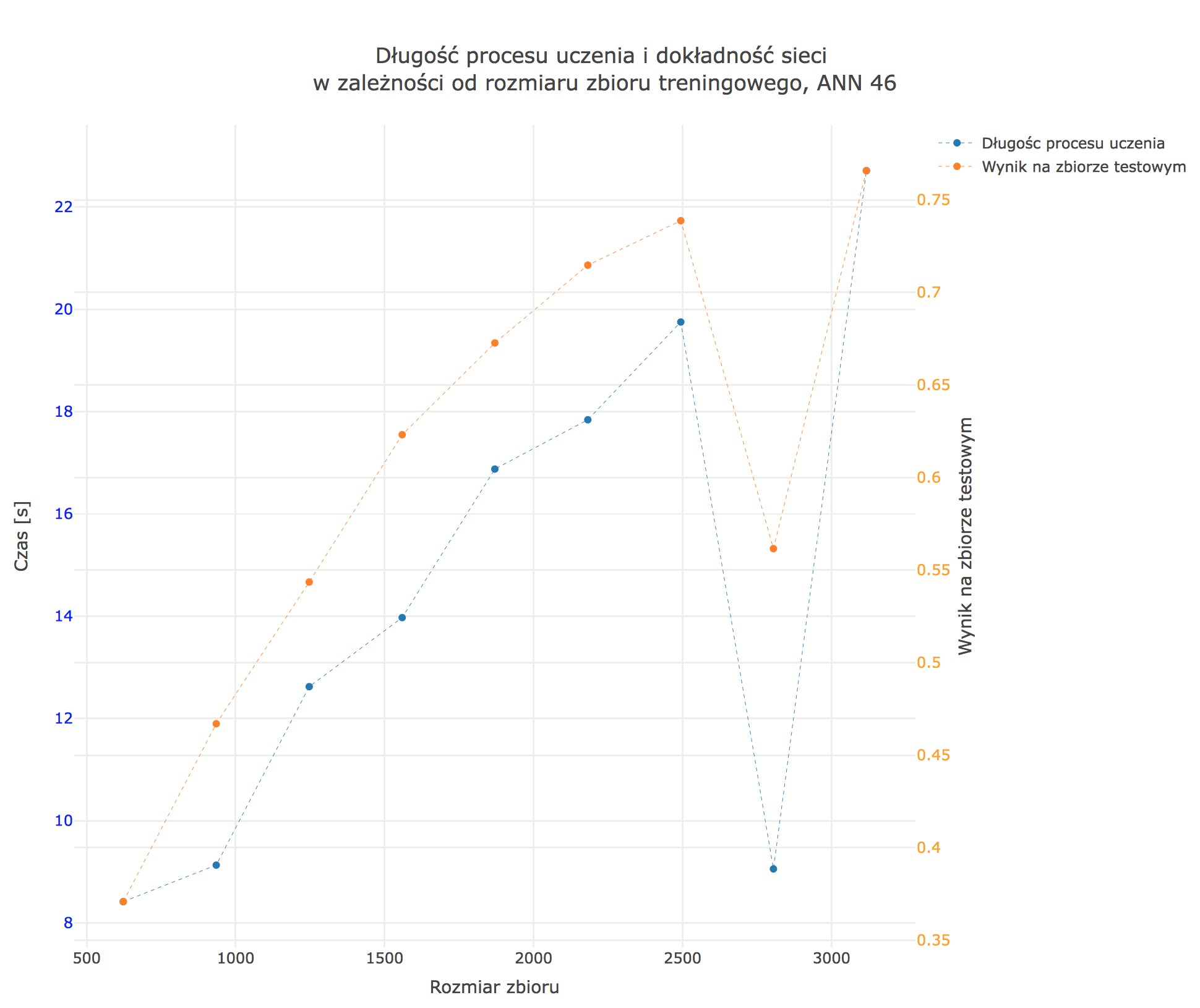
Poniższy wykres wykorzystuje powszechnie stosowane do oceny klasyfikatorów krzywe ROC. Do jego utworzenia konieczne było sprawdzenie jakości obu sieci (dla wejścia z modelu wykorzystującego 46 i 68 punktów charakterystycznych twarzy) dla różnych wartości progu klasyfikacyjnego. Każdy punkt na krzywej odpowiada współczynnikom False Positive Rate i True Positive Rate uzyskanym w wyniku ustalenia pewnego progu klasyfikacyjnego. Im krzywa dla danego modelu bliższa jest wykresowi prostej *y* = *x*, tym bliższy jest on klasyfikatorowi losowemu. Z kolei im bliżej lewej i górnej krawędzi, tym bliżej jest on klasyfikatora idealnego (reprezentowanego przez punkt (0*,*1)).



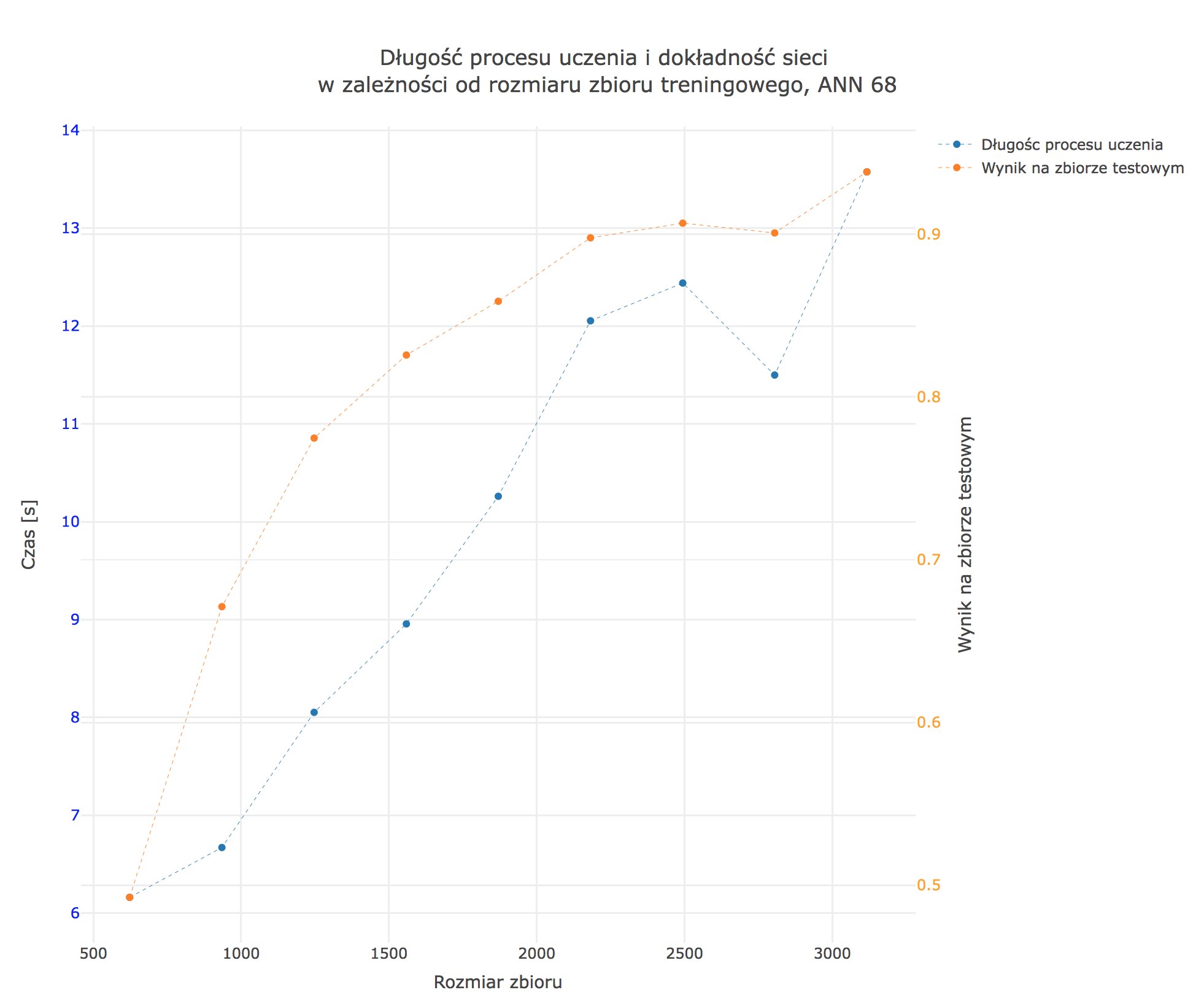
Rysunek 27: Krzywe ROC (Receiver Operating Characteristic) dla sieci neuronowych

Z powyższego wykresu łatwo można odczytać, że wykorzystanie 68 punktów charakterystycznych daje znacznie lepsza czułość i specyficzność klasyfikatora. Optymalny punkt odcięcia (próg klasy- fikacji) na postawie krzywej możemy dla tego modelu oszacować na około 0*.*90 (wiemy to, ponieważ przy wyliczaniu współczynników TPR i FPR zapisywane były progi klasyfikacji).

Do utworzenia poniższych dwóch wykresów należało w sposób losowy wybierać odpowiednie podzbiory zbioru treningowego, którego liczność wynosi około 3100 zdjęć. Następnie mierzony był czas potrzebny na trening oraz wynik osiągany na zbiorze testowym (w sensie próbek był on rozłączny ze zbiorem testowym).



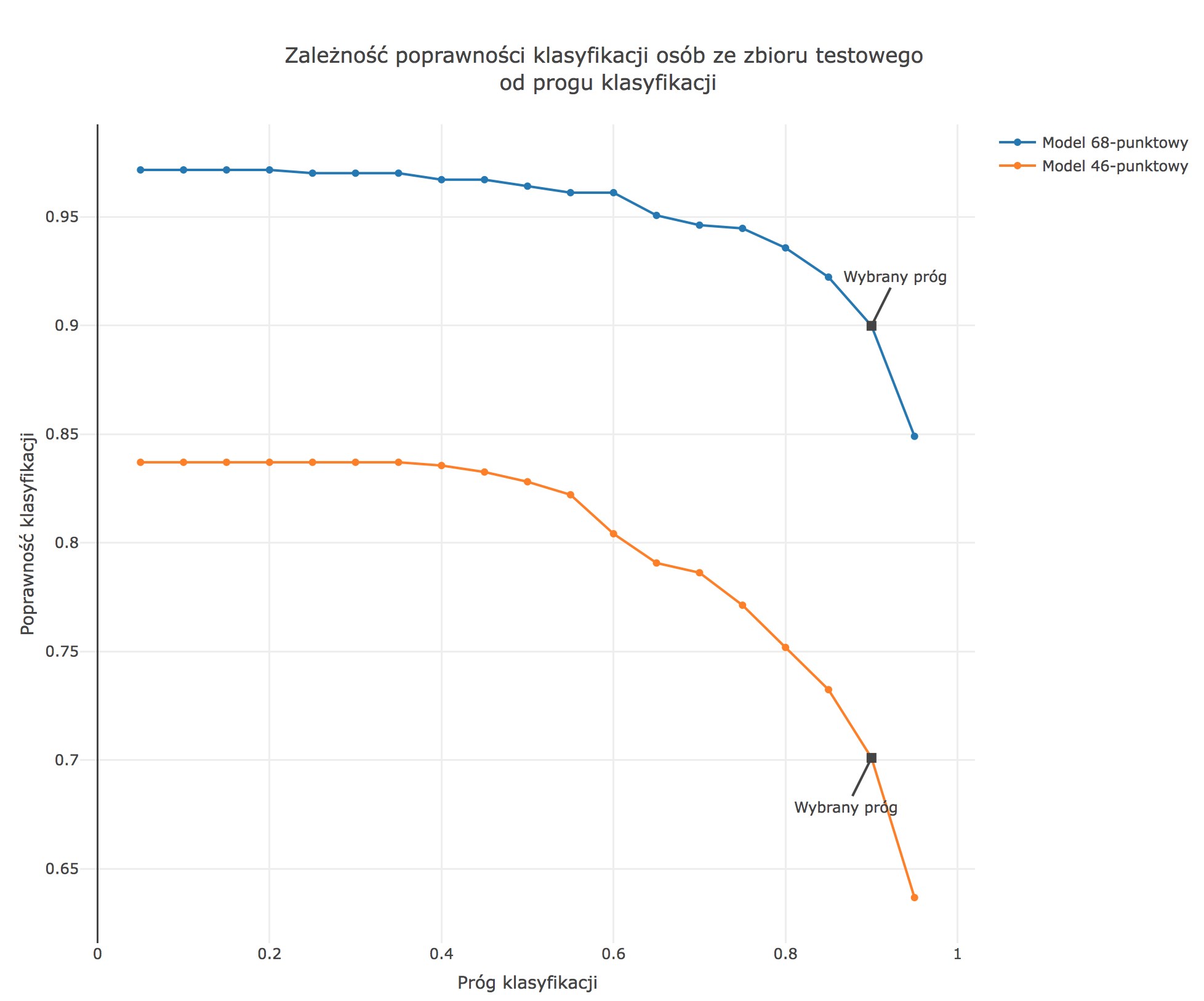
Rysunek 28: Czas uczenia i wynik metody ANN w zależności od rozmiaru zbioru treningowego dla modelu 46-punktowego



Rysunek 29: Czas uczenia i wynik metody ANN w zależności od rozmiaru zbioru treningowego dla modelu 68-punktowego

Dwa powyższe wykresy pozwalaj wysnuć istotny wniosek - skoro sieć neuronowa wykorzystująca model 68-punktowy osiąga zbieżność znacząco szybciej niż w przypadku modelu 46-punktowego, to dodatkowe punkty niosące sobą kluczowe dla identyfikacji informacje. Pozostałe obserwacje są natychmiastowe - im większa liczność zbioru treningowego, tym lepsze wyniki osiągane na zbiorze testowym, jednak rośnie tez czas treningu (czas potrzebny na osiągniecie zbieżności lub wykonanie 300 iteracji w przypadku gdy zbieżność nie zostanie osiągnięta. Zbieżność osiągana jest w przypadku niepoprawienia wartości funkcji straty w kolejnych dwóch iteracjach o co najmniej pewien epsilon, mający w naszym systemie wartość 0*.*0001).

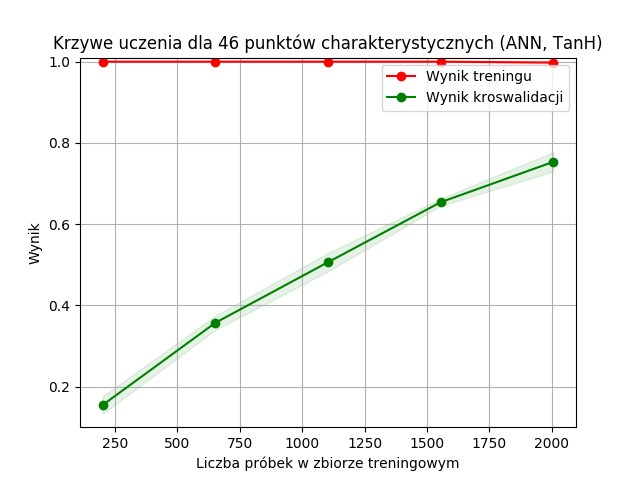
Poniższy wykres wymagał przetestowania wpływu wyboru progu klasyfikacji na wyniki testów danego modelu.



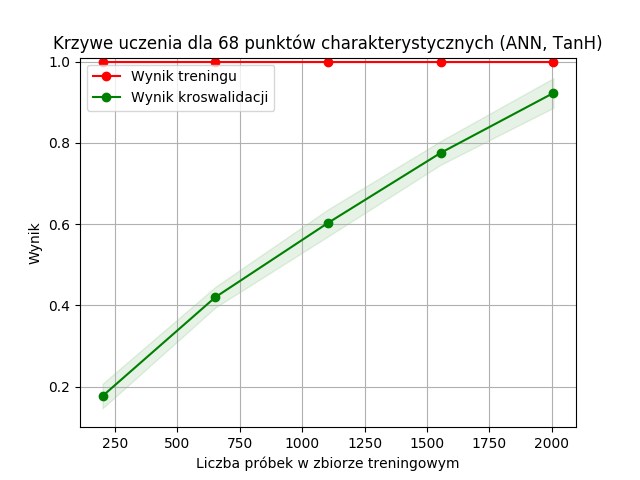
Rysunek 30: Zależność poprawności klasyfikacji osób ze zbioru testowego od progu klasyfikacji

Im wyższy próg klasyfikacji, tym więcej mamy błędnych klasyfikacji, tzw. false negative - zwracana jest etykieta „unclassified”, mimo że dana próbka należy do zbioru osób znanych klasyfikatorowi. Widoczna jest także różnica miedzy modelem 68- a 46-punktowym, która powiększa się wraz ze wzrostem progu klasyfikacji. Należy zwrócić uwagę, że nawet dla wysokich warto sic tego parametru (bliskich 1) poprawność oscyluje na zadowalającym poziomie.

Poniższe dwa wykresy przedstawiają krzywe uczenia i wykorzystują wbudowane w bibliotekę scikitlearn mechanizmy oceny klasyfikatora za pomoc potrójnej kroswalidacji. Kroswalidacja jest metoda statystyczna, w której zbiór treningowy dzielony jest na podzbiory - testowanie odbywa się kolejno na jednym z podzbiorów, a inne wykorzystywane są do uczenia. Zapobiega ona zjawisku przeuczenia. Średni wynik zaznaczony jest na wykresie za pomoc punktu, a przez zacienienie oznaczono obszar jednego odchylenia standardowego.



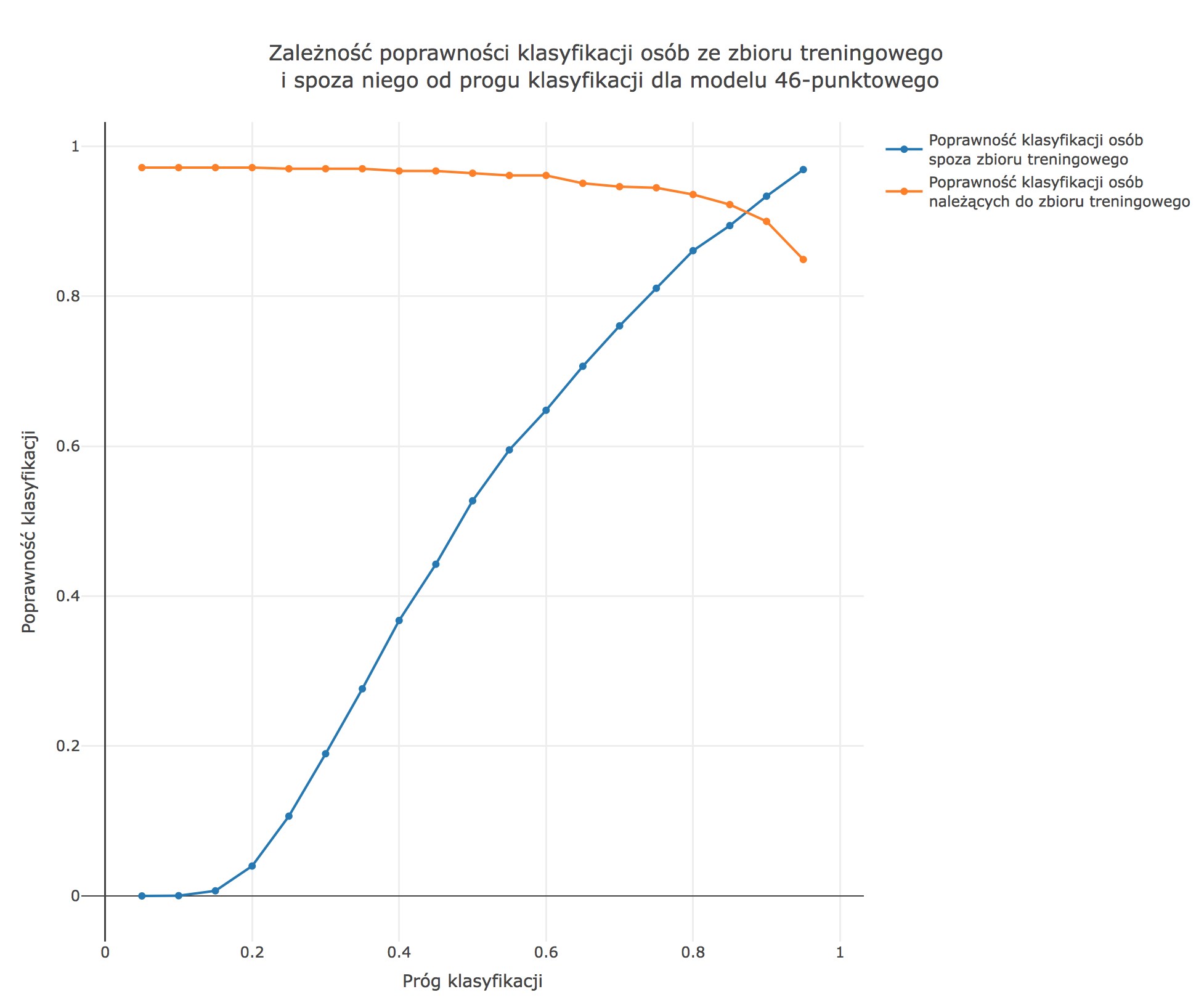
#### Rysunek 31: Krzywe uczenia dla sieci neuronowych, model 46-punktowy



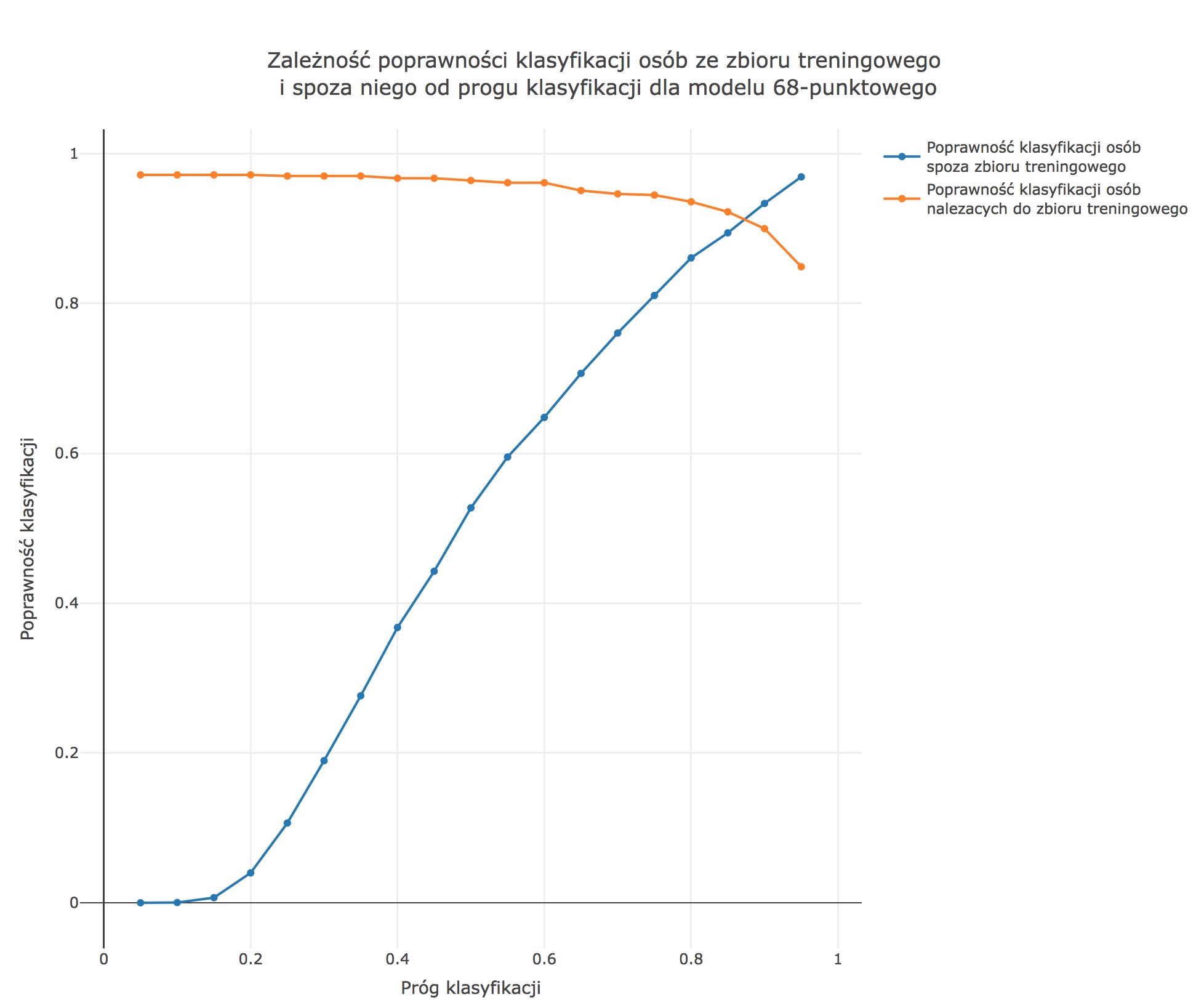
Rysunek 32: Krzywe uczenia dla sieci neuronowych, model 68-punktowy

Kolejny raz okazuje się, że model 68-punktowy osiąga wyraźnie lepszy wynik od 46-punktowego. Zgodnie z przewidywaniami na podstawie poprzednich rozważań, wynik kroswalidacji ro snie wraz ze wzrostem liczby próbek w zbiorze treningowym.

Aby poprawnie wyznaczyć domy silne warto sic progu klasyfikacji, należało porównać poprawność metody ANN dla zbioru walidacyjnego i zbioru osób spoza zbioru treningowego (a wiec takich, dla których system powinien zwrócić etykietę „unclassified”) dla zmiennego progu:



Rysunek 33: Zależność poprawnosic klasyfikacji osób ze zbioru walidacyjnego oraz osób które nie należą do zbioru treningowego od progu klasyfikacji dla modelu 46-punktowego.



Rysunek 34: Zależność poprawność klasyfikacji osób ze zbioru walidacyjnego oraz osób które nie należą do zbioru treningowego od progu klasyfikacji dla modelu 68-punktowego.

Dla obu modeli możemy próg ustalić na poziomie około 0*.*90.

## 8.2 Wyniki metody wektorów wspierających

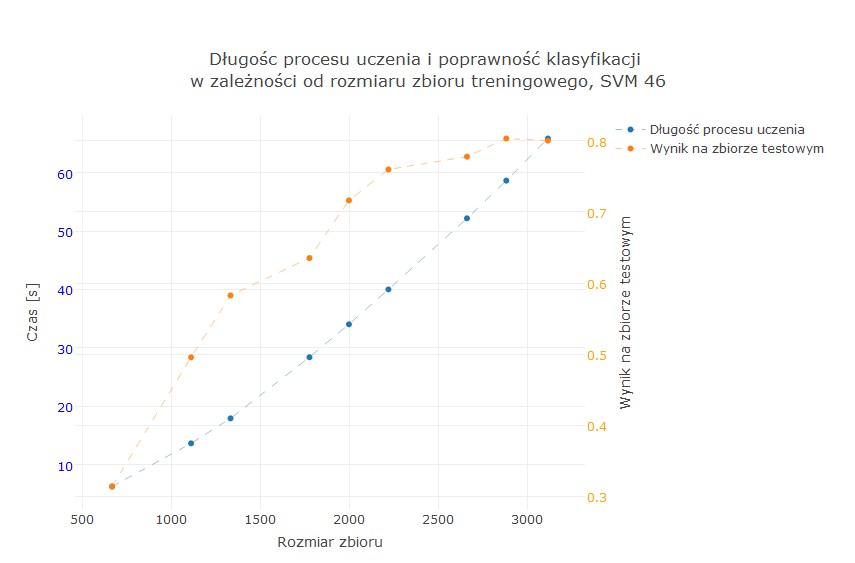
Podobnie jak w przypadku poprzedniej metody, administrator systemu może modyfikować wybrane parametry metody. Możliwe do edycji parametry opisane zostały w instrukcji administratora, która znajduje się w rozdziale 7.7.1.

Do implementacji tej metody zostały wykorzystane klasy SVC oraz OneVsRestClassifier z biblioteki scikit-learn. Do kalibracji prawdopodobieństwa wykorzystana została klasa CalibratedClassifierCV, również z biblioteki scikit-learn. Optymalne parametry zostały znalezione za pomocą klasy GridSearchCV z tej samej biblioteki. Znalezione wartości dla metody wektorów wspierających (trenowanej na zbiorze treningowym stanowiącym 70% sumy faces94 i faces95) przedstawia lista poniżej:

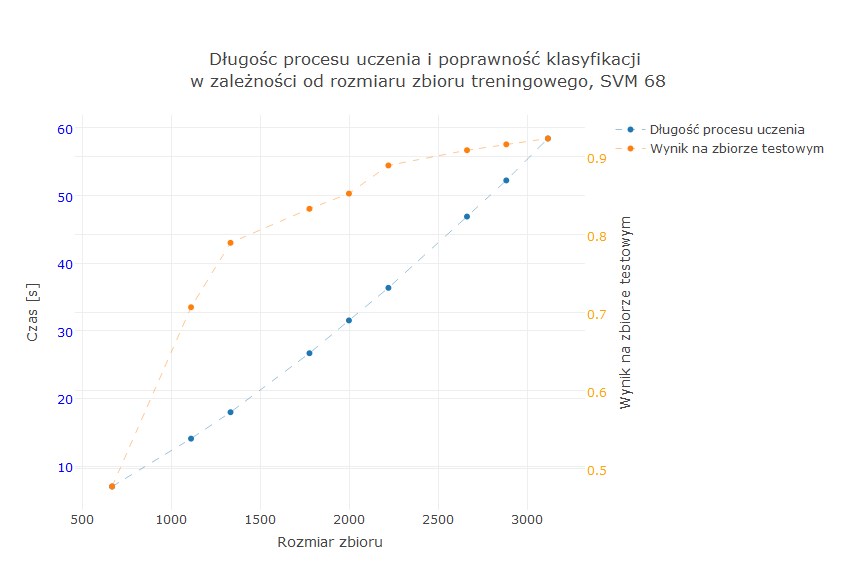
* Funkcja jądrowa: 0*rbf*0
* Parametr funkcji jądrowej *γ*:
  + model 46-punktowy: 0*.*00024
  + model 68-punktowy: 0*.*0039
* Parametr *C*:
  + model 46-punktowy: 8192
  + model 68-punktowy: 64

Ponadto wartość progu klasyfikacyjnego ustalono na 0*.*6 dla modelu 46-punktowego, oraz 0*.*7 dla modelu 68-punktowego. Poniżej tej wartości prawdopodobieństwa klasyfikacji zwracana jest etyki- eta „unclassified”.

Pierwsza ważna własnością która warto rozważyć, jest czas jaki należy przeznaczy c na nauczenie klasyfikatora danych treningowych lub, inaczej mówiąc, wytrenowanie go. Na ponizszym wykresie przedstawiono czas treningu w zależności od rozmiaru danych. Klasyfikator był trenowany na podzbiorach całego zbioru treningowego. Po wytrenowaniu sprawdzana była poprawność klasy- fikacji na zbiorze testowym.



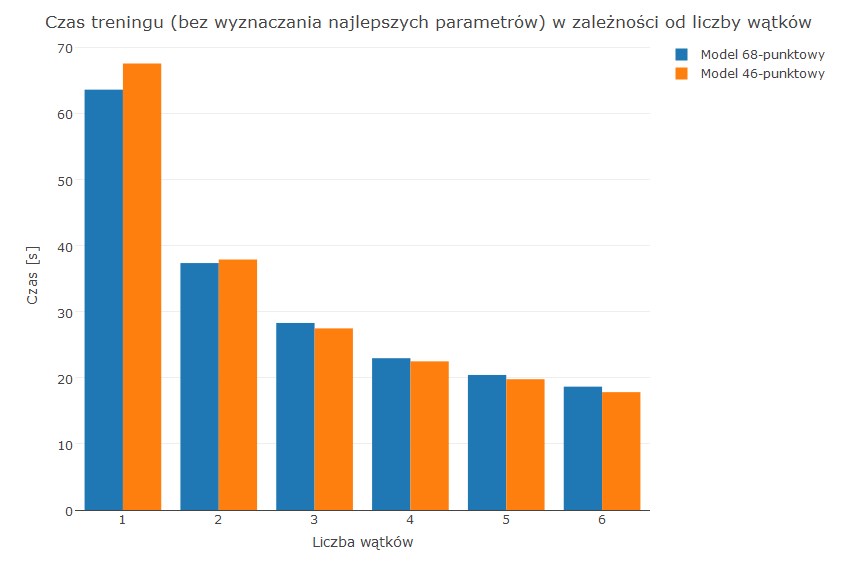
Rysunek 35: Czas uczenia i wynik metody SVM w zależności od rozmiaru zbioru treningowego dla modelu 46-punktowego



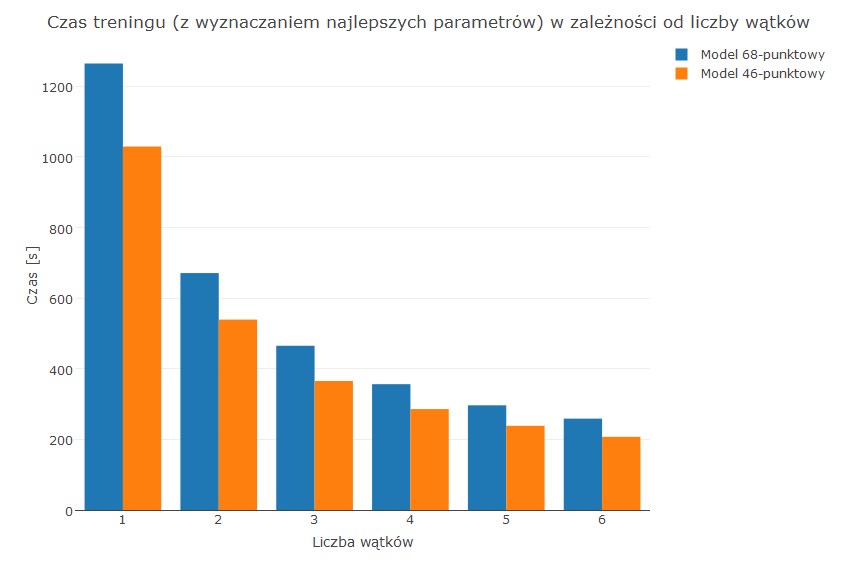
Rysunek 36: Czas uczenia i wynik metody SVM w zależności od rozmiaru zbioru treningowego dla modelu 68-punktowego

Z powyższych wykresów wynika, że czas treningu rośnie proporcjonalnie do rozmiaru zbioru treningowego, zarówno dla modelu 46-punktowego, jak i 68-punktowego. Większy zbiór treningowy poprawia jednak znacząco poprawność klasyfikacji. Jeśli chodzi o czas treningu, w tym przypadku nie widać dużych różnic miedzy dwoma modelami, jednak jest istotna różnica w poprawności klasyfikacji.

Ponieważ w problemie wieloklasowym należy wytrenować wiele niezależnych maszyn SVM, możemy to robić równolegle, a wiec warto sprawdzi c jak liczba wykorzystanych wątków wpływa na czas treningu. Przedstawiono to na poniższych wykresach.



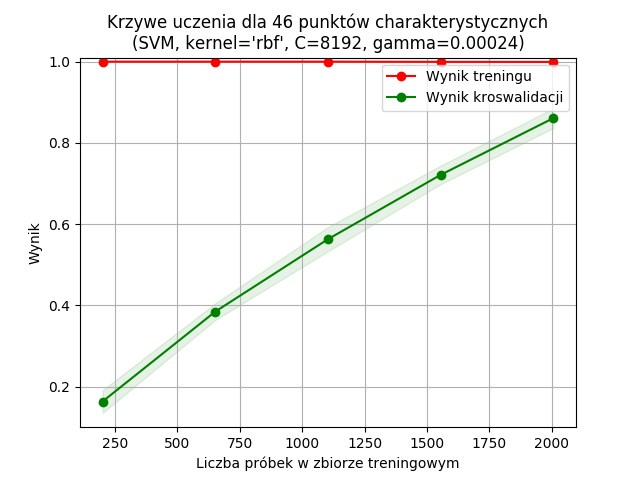
Rysunek 37: Czas uczenia (bez wyznaczania najlepszych parametrów) metody SVM w zależności od liczby wykorzystanych wątków



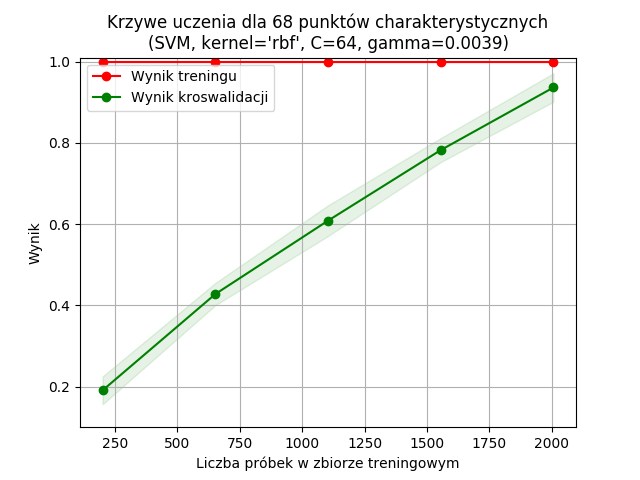
Rysunek 38: Czas uczenia (z wyznaczaniem najlepszych parametrów) metody SVM w zależności od liczby wykorzystanych wątków

Wykorzystanie wielu wątków pozwala znacząco skrócić czas uczenia, co jest szczególnie przydatne, gdy szukamy najlepszych parametrów. W tym przypadku dużo zależy od liczby kombinacji tych właśnie parametrów. Dla domy silnych danych w aplikacji, które posłużyły do wygenerowania powyższych wykresów, tych kombinacji było 99. Na wykresie 38 zauważamy również wyraźna różnice pomiędzy dwoma modelami jeśli chodzi o czas treningu. Przy użyciu modelu 46-punktowego udało się zmniejszyć czas treningu o prawie 19% w porównaniu do modelu 68-punktowego.

Na kolejnych wykresach przedstawiono krzywe uczenia, które pokazuj jak ilość dostępnych danych wpływa na poprawność klasyfikacji.



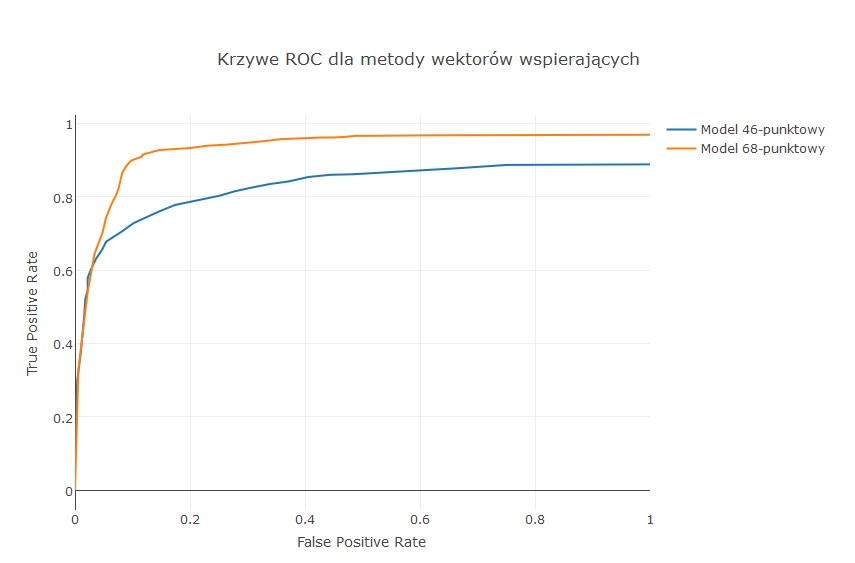
### Rysunek 39: Krzywe uczenia dla metody wektorów wspierających, model 46-punktowy



### Rysunek 40: Krzywe uczenia dla metody wektorów wspierających, model 68-punktowy

Tak jak można było wnioskować z poprzednich wykresów, więcej danych pozytywnie wpływa na poprawność klasyfikacji. Warto tutaj zwróci c uwagę na wynik treningu. Ponieważ wykorzystano odpowiednio duże wartości parametru *C*, to wszystkie punkty ze zbioru treningowego zaklasyfikowano poprawnie.

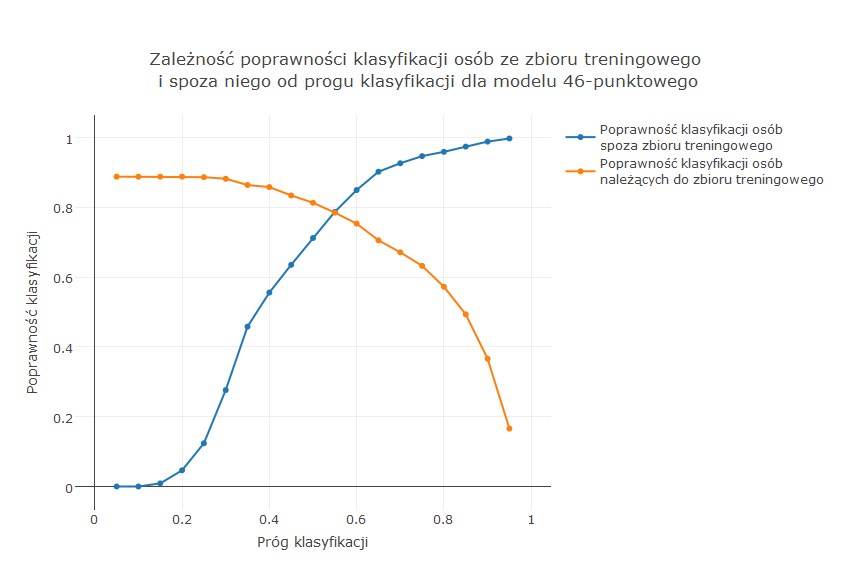
Podobnie jak w przypadku sztucznych sieci neuronowych, do oceny tej metody również możemy wykorzystać krzywe ROC, które przedstawiono na poniższym wykresie.



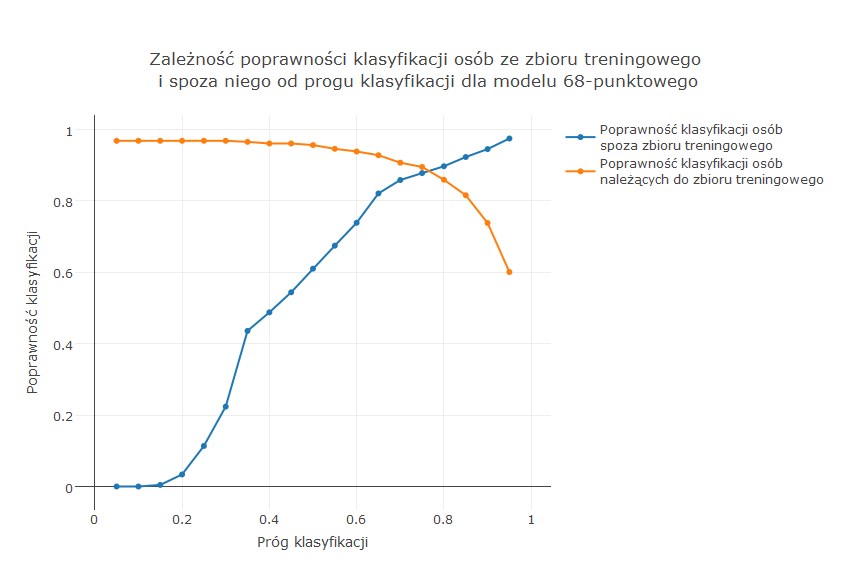
Rysunek 41: Krzywe ROC (Receiver Operating Characteristic) dla metody wektorów wspierających

Po raz kolejny można zauważyć, że lepsze wyniki osiągamy wykorzystując model 68-punktowy. Metoda SVM sprawdza się bardzo dobrze, gdy użyjemy jej do klasyfikacji osób, których zdjęcia mamy w zbiorze treningowym. Dla takich osób uzyskujemy wysoka oprawność klasyfikacji, jednak False Positive Rate również osiąga wtedy wysoka wartość. Dla małych warto sic False Positive Rate niestety znacząco maleje również True Positive Rate.

W ustawieniach aplikacji można modyfikować próg klasyfikacji, należało wiec wyznaczyć pewna wartość domy slna tego progu. W tym celu wyznaczono poprawność klasyfikacji na zbiorze walidacyjnym, a także na zbiorze osób, które nie należą do zbioru treningowego (a wiec nie powinny zostać zaklasyfikowane). Otrzymane wyniki przedstawiono na poniższych wykresach.



Rysunek 42: Zależność poprawna sieci klasyfikacji osób ze zbioru walidacyjnego oraz osób, które nie należą do zbioru treningowego od progu klasyfikacji dla modelu 46-punktowego.

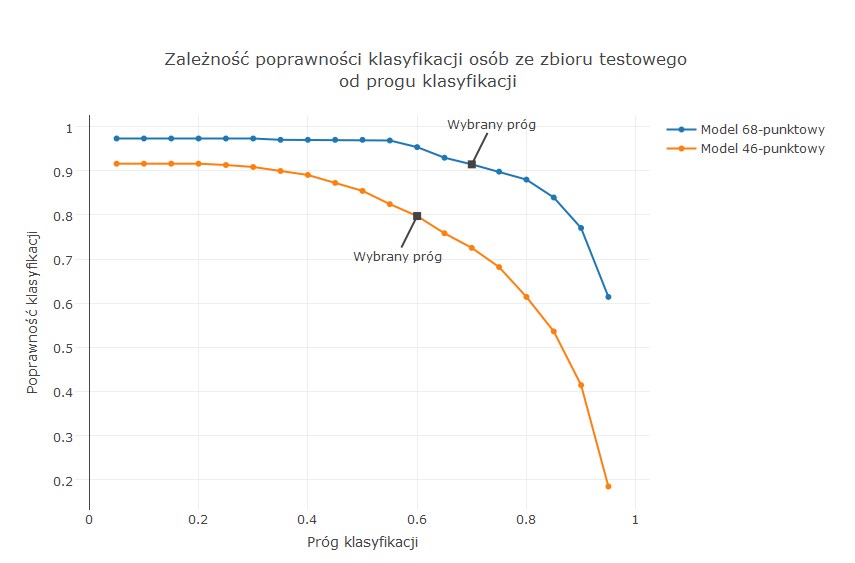


Rysunek 43: Zależność poprawna sieci klasyfikacji osób ze zbioru walidacyjnego oraz osób, które nie należą do zbioru treningowego od progu klasyfikacji dla modelu 68-punktowego.

Na podstawie powyższych wykresów, dla modelu 46-punktowego, jako próg klasyfikacji wybrano wartość 0*.*6, gdyż zapewnia zadowalająca poprawność na obu zbiorach (około 0*.*8). Podobnie dla modelu 68-punktowego, dla którego próg ustalono na 0*.*7, który to zapewnia poprawność na poziomie

0*.*9 na dwóch zbiorach, dla których przygotowano wykresy.

Ostatnia, ale również ważna cecha która należy rozważyć, jest poprawność na zbiorze testowym. Na poniższym wykresie przedstawiono wyniki osiągnięte przy użyciu obu modeli w zależności od progu klasyfikacji, z wyróżnionymi wybranymi wcześniej domy slnymi progami.



Rysunek 44: Zależność poprawna sieci klasyfikacji osób ze zbioru testowego od progu klasyfikacji dla metody wektorów wspierających

Ponownie model 68-punktowy pozwala na uzyskanie znacząco lepszej poprawności klasyfikacji. Różnice są szczególnie widoczne dla wyższych wartości progu klasyfikacji (powyżej 0*.*6). Oznacza to, prawdopodobieństwa klasyfikacji uzyskane za pomoc modelu 46-punktowego są niższe niż te, które otrzymano przy użyciu modelu 68-punktowego. Z powyższego wykresu wynika również, że progi klasyfikacji wybrane, jako domyślne dają dobra poprawność (około 0*.*8 i 0*.*9, odpowiednio dla modelu 46-punktowego i 68-punktowego). Niższe progi klasyfikacji pozwoliłyby uzyskać jeszcze lepsza poprawność, jednak należy pamiętać, że spowodowałyby również powstanie większej liczby błędów dla osób spoza zbioru treningowego, jak pokazano na wykresach 42 oraz 43.

## 8.3 Wyniki programowania genetycznego

Moduł programowania genetycznego pozwala (podobnie jak dwa, wspomniane wcześniej moduły ANN i SVM) na określanie hiper parametrów oraz ich wyszukiwania. Sposób konfiguracji został przedstawiony w instrukcji w rozdziale 7.7.1.

Do implementacji metody wykorzystano klasę SymbolicClassifier, która jest została zbudowana na bazie biblioteki gplearn. Ponadto, do znalezienia optymalnych hiper parametrów wykorzystano klasę GridSearchCV z biblioteki scikit-learn.

Ze względu na mnogosc możliwe sic parametrów, do znalezienia optymalnych wartości przyjęliśmy pewne stałe parametry, które maja niewielki wpływ na jakość znalezienia optymalnego klasyfikatora, a jedynie mogłyby wpłynąć na czas lub sposób ich znalezienia. Do tych parametrów należą:

* rozmiar populacji (“population\_size“): 100;
* rozmiar elity (“elite\_size“): 5;
* maksymalna liczba generacji (“generations“): 500;
* kryterium stopu dla funkcji dopasowania (“stopping\_criteria“): 1.0;

Inna część parametrów została ustawiona w sposób arbitralny, wynikający ze specyfiki problemu oraz ograniczeń sprzętowych. Te parametry to miedzy innymi:

* minimalna i maksymalna głębokość programu ( “depth“): [2, 25] - zapobiega to powstawaniu długich programów, które mogłyby wyczerpać dostępna pamięć urządzenia;
* funkcja dopasowania (“metric“): “accuracy“ - wynika to z optymalizacji klasyfikatora pod katem rozpoznawania poszczególnych osób;
* metoda generacji populacji początkowej (“init\_method“): “full“ - dobrana ze względu na duży wymiar przestrzeni zmiennych wejściowych (pozwala na użycie większej ilość zmiennych);
* prawdopodobieństwo wykonania operacji *crossover* (“p\_crossover“): 0.8;
* prawdopodobieństwo wykonania operacji *mutacja poddrzewa* (“p\_subtree\_mutation“): 0.2;
* prawdopodobieństwo wykonania operacji *dzwignia* (“p\_hoist\_mutation“): 0.0 - operacja nie jest wykorzystywana;
* prawdopodobieństwo wykonania operacji *mutacja punktowa* (“p\_point\_mutation“): 0.0 - operacja nie jest wykorzystywana;
* prawdopodobieństwo wykonania operacji *zamiana punktowa* (“p\_point\_replace“): 0.0 - operacja nie jest wykorzystywana;

Hiperparametry, które zespół zdecydował się wybrać do optymalizacji algorytmu programowania genetycznego to:

* rozmiar puli turniejowej (“tournament\_size“): 5, 10 lub 15;
* strategia selekcji programów do ewolucji (“generations“): “roulette“, “rank“ lub “tournament“;
* minimalna i maksymalna wartosc stałych terminali ( “const\_range“): [-1.0, 1.0], [-2.5, 2.5] lub [-5.0, 5.0],
* minimalna i maksymalna głebokosc programu z populacji poczatkowej ( “init\_depth“): [6, 8], [7, 9] lub [8, 10];
* zbiór dostępnych funkcji w programach (“function\_set“): [“add“, “sub“, “min“, “div“] lub [“add“, “sub“, “min“, “div“, “sqrt“, “min“, “max“, “if“]
* współczynnik kary za długosc ( “parsimony\_coefficient“): 0, 0.000001, 0.00001 lub 0.0001;

Po zastosowaniu algorytmu optymalizujacego hiperparametry okazało się, że najoptymalniejsze wartosci powyzszych hiperparametrów sanastepujace:

* rozmiar puli turniejowej (“tournament\_size“): 5;
* strategia selekcji programów do ewolucji (“generations“): “roulette“;
* minimalna i maksymalna wartosc stałych terminali ( “const\_range“): [-2.5, 2.5];
* minimalna i maksymalna głebokosc programu z populacji poczatkowej ( “init\_depth“): [6, 8];
* zbiór dostępnych funkcji w programach (“function\_set“): [“add“, “sub“, “min“, “div“,

“sqrt“, “min“, “max“, “if“]

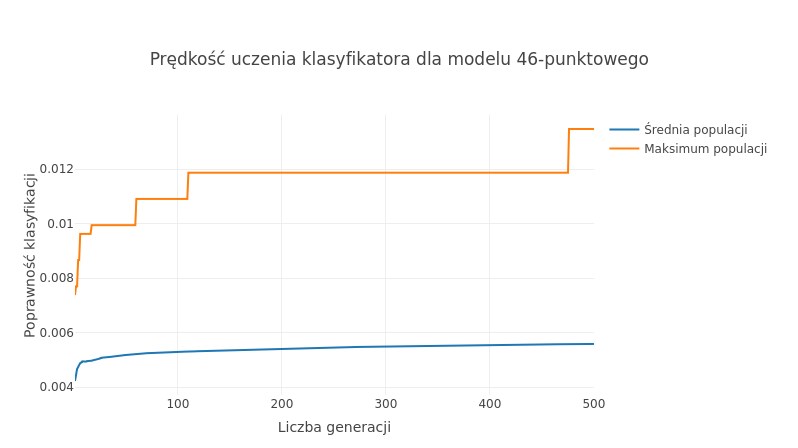
* współczynnik kary za długosc ( “parsimony\_coefficient“): 0;

Należy pamiętać, że programowanie genetyczne jest metoda korzystającą w dużym stopniu z losowo sic generacji początkowej, losowości doboru programów do wykonania operacji genetycznych, jak również wyboru samej operacji genetycznej. W związku z czym, znalezione optymalne hiperparametry mogą ulec zmianie w zależności od stanu generatora liczb pseudolosowych - jest to konsekwencja wspomnianej cechy metody.

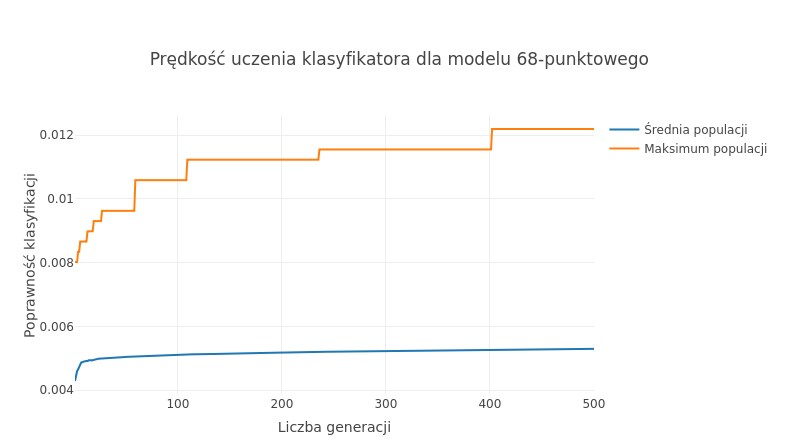
Interpretacja wyników znalezionych programów opiera się na dyskretyzacji przestrzeni możliwych wyników do zbioru szukanych klas. Polega to na znalezieniu wartości minimalnej i maksymalnej, a następnie podzieleniu przedziału miedzy tymi wartościami na *N* podprzedziałów. Każdy z nich odpowiada konkretnej klasie - jeżeli program, na podstawie danych wejściowych, zwrócił wartość z *i*-tego przedziału, to przyjmujemy, że wynikiem klasyfikacji jest *i*-ta klasa.

W konsekwencji, oczekujemy, że w procesie uczenia będziemy preferowali programy, które dla pierwszych klas (w ustalonym porządku) Beda zwracać niższe warto sic, za s w miarę zwiększania indeksu klasy będą rosły wartości zwracane przez program, który służyłby za klasyfikator.

W celu zbadania skuteczności tej metody klasyfikacji, przeanalizujemy zależano sc jako sic klasy- fikacji od ilości generacji (czyli procesów tworzenia nowych populacji) oraz od wielko sic zbioru treningowego.

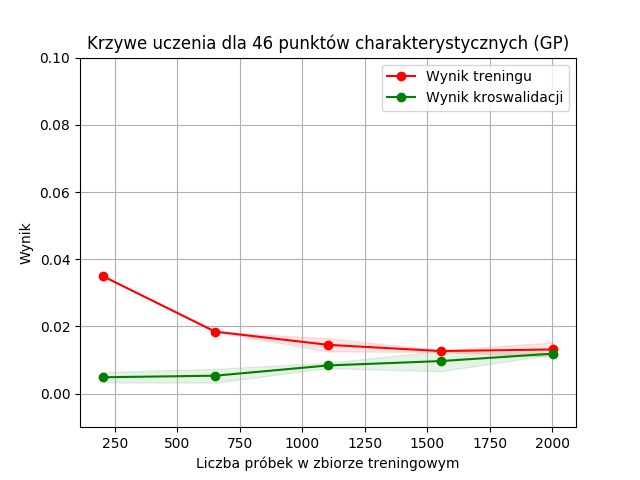


Rysunek 45: Prędkość uczenia się klasyfikatora metoda programowania genetycznego dla modelu 46-punktowego

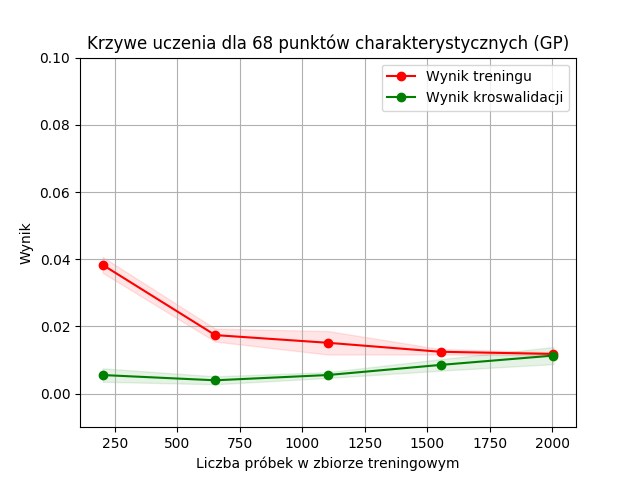


Rysunek 46: Prędkość uczenia się klasyfikatora metoda programowania genetycznego dla modelu 68-punktowego

Powyższe wykresy prezentują szybkość nauki klasyfikatora dla 500 generacji. Jak widać, niezależnie od dokonania wyboru modelu (pomiędzy 46-punktowym a 68-punktowym) wraz ze wzrostem liczby generacji, poprawa jakości klasyfikacji zachodzi o wiele wolniej. Warto zauważyć, że krzywa przedstawiająca średnia jakość klasyfikacji dla całej populacji kształtem przypomina krzywa logarytmiczna. Co więcej, wraz z kolejnymi populacji głębokość programów do niej należących ulega zwiększeniu, co z kolei skutkuje wydłużeniem realnego czasu ewaluacji dopasowania poszczególnych programów.



### Rysunek 47: Krzywe uczenia dla metody programowania genetycznego, model 46-punktowy



### Rysunek 48: Krzywe uczenia dla metody programowania genetycznego, model 68-punktowy

Widzimy, że podobnie jak w przypadku pozostałych metod, większa liczba próbek wpływa na jakość klasyfikacji testowej. Obserwujemy jednak spadek jakość klasyfikacji na zbiorze treningowym. Co więcej, odbiega ona znacznie od wyników z poprzednich metod, które potrafiły uzyskiwać warto sic zbliżone do 100%. Możemy wiec stwierdzić, że poniższa metoda klasyfikacji służącej identyfikacji osób na podstawie zdjęć nie sprawdza się dla użytych zbiorów zdjęć.

## 8.4 Porównanie metod i wnioski końcowe

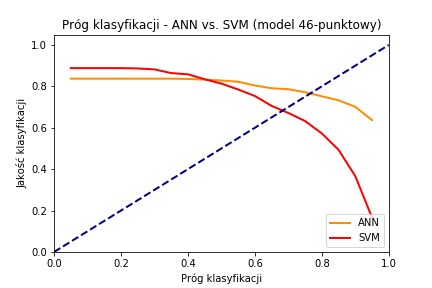
Najlepsza metoda klasyfikacji osób na podstawie zdjęć okazały się sztuczne sieci neuronowe. Siec o odpowiednich parametrach daje możliwość ustawienia wysokiego progu klasyfikacji, w związku z czym współczynnik False Positive Rate jest niższy w porównaniu z pozostałymi metodami. W przypadku rozważań pomijających dane spoza zbioru treningowego (np. wykorzystujących jedynie zbiór testowy, a wiec nie uwzgledniających False Positive Rate) metody ANN i SVM osiągają podobne wyniki.

Zaleta opracowanej w ramach pracy inżynierskiej implementacji metody wektorów wspierających jest możliwość treningu klasyfikatora na kilku watkach, co znacznie skraca czas procesu uczenia w wyniku wykorzystania pełnej mocy obliczeniowej nowoczesnego procesora. Mimo wykorzystania jednego wątku, czas treningu sztucznych sieci neuronowych jest porównywalny i wynosi kilkanaście sekund.

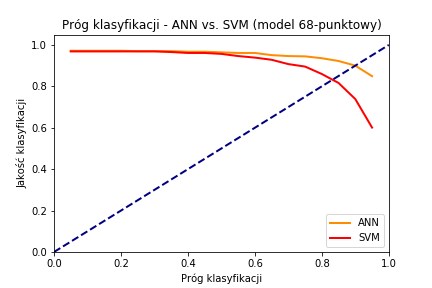
W wyniku rozważań i eksperymentów z metoda programowania genetycznego okazało się, że metoda ta daje niesatysfakcjonujące wyniki zarówno jeśli chodzi o czas treningu jak i testy wydajnościowe na przygotowanym zbiorze testowym.

Porównanie wybranych modeli twarzy: 46- i 68-punktowego wypada korzystniej na rzecz tego drugiego - klasyfikatory ANN i SVM osiąga ja znacząco lepsze wyniki mając do dyspozycji bardziej deskryptywne wejście. Ponadto metoda ANN uzyskuje zbieżna sc szybciej dla wyżej wymiarowego wejścia. Dla SVM czas treningu dla 68-punktowego wejścia jest dłuższy, jednak testy poprawna sic klasyfikacji na wybranych zbiorach uzasadniają wybór tego modelu. Dla metody GP otrzymane wyniki były na tyle niezadowalające, że różnice miedzy dwoma modelami twarzy można uznać za zaniedbywalne.

Dwa poniższe wykresy maja na celu pokazanie zależności miedzy wyborem progu klasyfikacji a jakością klasyfikacji (mierzona jako stosunek poprawnych identyfikacji do wszystkich zleconych klasyfikacji) na zbiorze testowym:

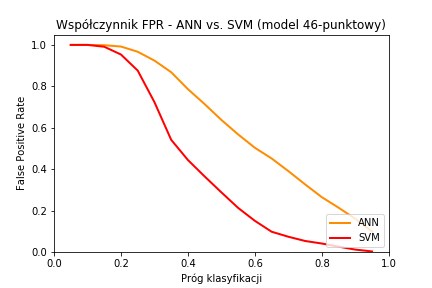


Rysunek 49: Porównanie wpływu progu klasyfikacji na wynik ANN i SVM (dla modelu 46punktowego)

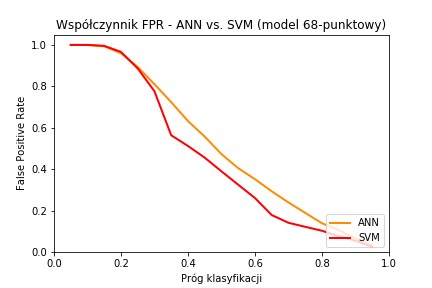


Rysunek 50: Porównanie wpływu progu klasyfikacji na wynik ANN i SVM (dla modelu 68punktowego)

Z analizy wynika, że klasyfikator ANN uzyskuje dość dobre wyniki dla dużego zakresu warto sic progu klasyfikacji, w przeciwieństwie do SVM, który osiąga zdecydowanie niższa jakość klasyfikacji dla wysokich progów. Obydwa algorytmy działają lepiej dla modelu twarzy wykorzystującego 68 punktów charakterystycznych. O tym jak wybór progu wpływa na błędy pierwszego rodzaju mówią dwa kolejne wykresy:



Rysunek 51: Porównanie współczynnika False Positive Rate dla klasyfikatorów ANN i SVM (dla modelu 46-punktowego)



Rysunek 52: Porównanie współczynnika False Positive Rate dla klasyfikatorów ANN i SVM (dla modelu 68-punktowego)

Uwagę zwraca duża przewaga klasyfikatora SVM dla modelu 46-punktowego - dla 68-punktowego różnice nie są aż tak zauważalne. Należy jednak mieć na uwadze, że z wcześniejszych rozważa n wynika, że dla ANN możemy ustalić znacząco wyższy próg klasyfikacji niż dla SVM, a wówczas klasyfikator ten uzyskuje odsetek błędów pierwszego rodzaju na poziomie podobnym, jak metoda wektorów wspierających.

# 9 Podsumowanie

W niniejszej *pracy inżynierskiej* zawarto teoretyczny wstęp zawierający opis trzech metod wykorzystanych podczas opracowywania aplikacji do rozpoznawania osób na podstawie zdjęć twarzy: programowania genetycznego, metody wektorów wspierających oraz sztucznych sieci neuronowych. Przedstawiono także istniejące aplikacje rozwiązujące to zagadnienie oraz zbiory danych wraz z przesłankami stojącymi za wyborem zbioru opracowanego przez Uniwersytet w Essex.

Zawarta jest także dokumentacja techniczna przedstawiająca wymagania funkcjonalne i niefunkcjonalne stawiane przed opracowywanym systemem informatycznym oraz jego komponenty wraz z zależnościami. W pracy znajdują się również skrócone instrukcje dla użytkownika i administratora systemu. W ramach pracy powstała aplikacja przeglądarkowa, której serwer składa się z trzech zasadniczych modułów identyfikacji, wykorzystujących wspomniane wyżej metody. Kod został przygotowany w sposób, który pozwala na łatwe rozszerzanie istniejącej aplikacji, co daje nadzieje na dalsza rozbudowę powstałej aplikacji. Dostarczone zostały uprzednio wytrenowane klasyfikatory gotowe do działania, jednak opracowany system udostępnia rozbudowane możliwości przetwarzania danych, dostosowywania opcji klasyfikatorów oraz ich ponownego treningu i eksportu.

Ostatnia część rozważa n stanowi przedstawienie wyników benchmarków i końcowe porównanie programowania genetycznego, metody wektorów wspierających i sztucznych sieci neuronowych w zadaniu identyfikacji osób na podstawie zdjęć twarzy. Okazało się, że algorytmy genetyczne nie dają zadowalających rezultatów. Zdecydowanie lepiej wypadają SVM i ANN, których czas treningu liczony jest w sekundach dla dużych zbiorów treningowych; osiągają one zadowalające wyniki dla parametrów takich jak jakość klasyfikacji i współczynnik błędów pierwszego rodzaju. Zgodnie z przewidywaniami, model twarzy oparty o 68 punktów charakterystycznych okazał się być bardziej efektywnym w zadaniu identyfikacji niż model 46-punktowy.

# Wykaz literatury

1. Krzysztof Slot, *Wybrane zagadnienia biometrii*, WKŁ 2008, Rozdział 6.3, strona 106: Automatyczne rozpoznawanie twarzy, 2008.
2. Nick McClure, *Tensorflow Machine Learning Cookbook*, Packt Publishing, 2017.
3. Cole Murray, *Building a Facial Recognition Pipeline with Deep Learning in Tensorflow, https://hackernoon.com/building-a-facial-recognition-pipeline-with-deep-learning-intensorflow-66e7645015b8* [Dostep: 01.10.2017]
4. Adam Geitgey, *Machine Learning is Fun! Part 4: Modern Face Recognition with*

*Deep Learning, https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-facerecognition-with-deep-learning-c3cffc121d78* [Dostep: 01.10.2017]

1. B. Amos, B. Ludwiczuk, M. Satyanarayanan, *„Openface: A general-purpose face recognition library with mobile applications”, CMU-CS-16-118, CMU School of Computer Science, Tech. Rep., 2016, http://reports-archive.adm.cs.cmu.edu/anon/anon/2016/CMU-CS-16-118.pdf* [Dostep: 01.10.2017]
2. Albiol A., Oliver J., Mossi J.M, *Who is who at different cameras: people re-identification using depth cameras. Computer Vision*, IET. Vol 6(5), pp. 378-387, 2012.
3. Adam Geitgey, *Image Classification Practical, http://www.di.ens.fr/willow/events/cvml2011/ materials/practical-classification/* [Dostep: 01.10.2017]
4. Davis E. King, *Dlib-ml: A Machine Learning Toolkit. Journal of Machine Learning Research 10, pp. 1755-1758, 2009, http://jmlr.csail.mit.edu/papers/volume10/king09a/king09a.pdf* [Dostep: 01.10.2017]
5. *Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011. http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html* [Dostep: 01.10.2017]
6. Bernd Heisele, Purdy Ho, Tomaso Poggio, *Face Recognition with Support Vector Machines: Global versus Component-based Approach, http://cbcl.mit.edu/publications/ps/iccv2001.pdf* [Dostep: 01.10.2017]
7. P. Jonathon Phillips, *Support Vector Machines Applied to Face Recognition, http://papers.nips.cc/paper/1609-support-vector-machines-applied-to-face-recognition.pdf* [Dostep: 01.10.2017]
8. Hani M. Ibrahem, Mohammed M. Nasef, Mahmoud Emam, *Genetic Programming based Face Recognition, International Journal of Computer Applications, Volume 69–No.27, May 2013* [Dostep: 01.10.2017]
9. Adam Pietrowcew, *Detekcja i rozpoznanie twarzy w obrazach cyfrowych: praca doktorska.*, Wydział EiTI PW, 2004.
10. *F.A.C.E. API, https://face-api.sightcorp.com/demo\_basic/* [Dostep: 01.10.2017]
11. *Betaface, https://betaface.com/demo.html* [Dostep: 01.10.2017]
12. *FaceRecognition, http://visagetechnologies.com/products-and-services/visagesdk/facerecognition/* [Dostep: 01.10.2017]
13. *Azure Face API, https://docs.microsoft.com/en-us/azure/cognitive-services/face/* [Dostep: 01.10.2017]
14. *Artificial neural network, https://en.wikipedia.org/wiki/Artificial\_neural\_network* [Dostep: 01.10.2017]
15. *Aproksymacja funkcji wielu zmiennych, Instytut Fizyki, Zakład Teorii Wzglednosci i As- trofizyki, Uniwersytet Jagiellonski, 2009, http://th.if.uj.edu.pl/ odrzywolek/homepage/presenta- tions/PL/Approx.pdf* [Dostep: 01.10.2017]
16. *A Beginner’s Guide To Understanding Convolutional Neural Networks, https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner%27s-Guide-ToUnderstanding-Convolutional-Neural-Networks/* [Dostep: 01.10.2017]
17. Michael Lones, *Genetic programming, http://www.macs.hw.ac.uk/ ml355/common/thesis/c6.html* [Dostep: 01.10.2017]
18. Matthew Walker, *Introduction to Genetic Programming, https://www.cs.montana.edu/ bwall/cs580/introduction\_to\_gp.pdf* [Dostep: 01.10.2017]
19. Diederik P. Kingma, Jimmy Ba, *Adam: A Method for Stochastic Optimization, https://arxiv.org/abs/1412.6980* [Dostep: 01.10.2017]
20. Libor Spacek , *Collection of Facial Images, http://cswww.essex.ac.uk/mv/allfaces/index.html* [Dostep: 01.10.2017]
21. *gplearn - implementation of Genetic Programming in Python, https://github.com/trevorstephens/gplearn* [Dostep: 01.10.2017]