Raport - Pożyczka

Adrian Politowski – 313128

1. Wczytywanie danych

Zaczniemy od wczytania danych oraz ich wyświetlenia:

	person_age	person_gender	...	previous_loan_defaults_on_file	loan_status
0	22.0	female	...	No	1
1	21.0	female	...	Yes	0
2	25.0	female	...	No	1
3	23.0	female	...	No	1
4	24.0	male	...	No	1
...	...	...	...	...	...
44995	27.0	male	...	No	1
44996	37.0	female	...	No	1
44997	33.0	male	...	No	1
44998	29.0	male	...	No	1
44999	24.0	male	...	No	1

[45000 rows x 14 columns]

Jak widzimy mamy 45 tyś rekordów

2. Sprawdzanie jakości oraz poprawa błędów

Przyjrzyjmy się dokładniej danym w tym celu wywołujemy info()

```
RangeIndex: 45000 entries, 0 to 44999
Data columns (total 14 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   person_age                      45000 non-null  float64
 1   person_gender                   45000 non-null  object
 2   person_education                45000 non-null  object
 3   person_income                   45000 non-null  float64
 4   person_emp_exp                  45000 non-null  int64
 5   person_home_ownership           45000 non-null  object
 6   loan_amnt                       45000 non-null  float64
 7   loan_intent                     45000 non-null  object
 8   loan_int_rate                   45000 non-null  float64
 9   loan_percent_income             45000 non-null  float64
 10  cb_person_cred_hist_length      45000 non-null  float64
 11  credit_score                    45000 non-null  int64
 12  previous_loan_defaults_on_file  45000 non-null  object
 13  loan_status                     45000 non-null  int64
dtypes: float64(6), int64(3), object(5)
```

Jak możemy zauważyć wiek oraz historia kredytowa jest w floacie. Nie potrzebujemy tego, dlatego zamieniamy na int. Dodatkowo możemy zauważyć, że nie występują tutaj braki danych dzięki kolumnie “Count”.

Możemy także przeanalizować wartości uzywając polecenia unique()

```
person_age:  [22 21 25 23 24 26 20 32 34 29 33 28 35 31 27 30 36 40 50 45 37 39 44 43
 41 46 38 47 42 48 49 58 65 51 53 66 61 54 57 59 62 60 55 52 64 70 78 69
 56 73 63 94 80 84 76 67]
person_gender:  ['female' 'male']
person_education:  ['Master' 'High School' 'Bachelor' 'Associate' 'Doctorate']
person_income:  [71948. 12282. 12438. ... 31924. 56942. 33164.]
person_emp_exp:  [ 0  3  1  5  4  2  7  6  8 12 10  9 14 13 11 15 16 17 19 28 25 18 24 22
 20 23 21 31 26 27 29 32 30 40 43 33 44 34 42 37 45 36 41 47 38 39 35 57
 46 49 48 50 76 62 61 58]
person_home_ownership:  ['RENT' 'OWN' 'MORTGAGE' 'OTHER']
loan_amnt:  [35000.  1000.  5500. ... 12229.  2771.  6665.]
loan_intent:  ['PERSONAL' 'EDUCATION' 'MEDICAL' 'VENTURE' 'HOMEIMPROVEMENT'
 'DEBTCONSOLIDATION']
loan_int_rate:  [16.02 11.14 12.87 ... 19.11 19.8  16.92]
loan_percent_income:  [0.49 0.08 0.44 0.53 0.19 0.37 0.35 0.13 0.34 0.3  0.27 0.25 0.05 0.24
 0.31 0.26 0.12 0.18 0.21 0.43 0.11 0.4  0.32 0.1  0.41 0.22 0.07 0.02
 0.17 0.51 0.38 0.33 0.06 0.03 0.28 0.04 0.09 0.2  0.23 0.15 0.48 0.5
 0.14 0.42 0.01 0.46 0.29 0.39 0.36 0.16 0.   0.57 0.45 0.52 0.55 0.61
 0.62 0.66 0.54 0.47 0.56 0.58 0.63 0.59]
cb_person_cred_hist_length:  [ 3  2  4  8  7  6  9 10  5 11 16 15 12 13 17 14 28 27 22 19 29 23 26 20
 21 30 25 24 18]
```

```
credit_score:  [561 504 635 675 586 532 701 585 544 640 621 651 573 708 583 670 663 694
 709 679 684 662 691 600 654 626 607 700 553 589 681 567 669 606 582 649
 602 616 631 637 695 620 622 645 624 570 648 652 559 623 609 579 688 661
 562 664 564 598 557 677 690 599 604 601 634 671 538 587 683 518 617 668
 673 706 536 689 595 584 642 614 597 625 603 643 508 505 593 686 646 697
 615 687 650 588 658 531 665 703 594 618 574 577 653 630 660 639 612 628
 592 580 678 672 613 566 718 484 699 656 659 636 554 578 674 608 569 629
 560 548 667 676 581 655 551 529 666 576 633 611 657 647 542 692 545 540
 525 537 641 539 563 712 491 590 572 528 638 627 596 547 507 565 693 522
 632 556 499 704 503 714 552 555 558 521 605 571 591 719 610 535 644 523
 546 702 711 534 682 447 725 680 568 524 710 707 619 460 696 527 511 477
 575 496 685 476 502 541 722 506 487 530 515 513 520 724 514 501 549 720
 486 716 465 509 517 444 482 698 443 512 526 454 550 462 717 727 715 543
 448 713 721 481 723 533 705 483 740 495 459 489 519 494 498 488 468 490
 479 510 472 480 421 473 500 493 441 485 450 467 461 475 463 728 735 458
 456 726 729 750 455 478 453 445 431 737 497 492 437 516 451 434 435 439
 466 440 449 469 471 730 470 734 751 739 736 738 733 732 731 741 748 744
 762 742 745 743 464 390 755 747 759 746 756 753 764 760 754 770 784 773
 765 772 419 474 430 418 420 768 457 446 767]
previous_loan_defaults_on_file:  ['No' 'Yes']
```

Istnieją obserwacje, których wiek wynosi 144 lata. Te obserwacje wydają się błędne, dlatego usuwamy takie obserwacje, których wiek jest większy niż 100. Tych obserwacji jest tylko 7.

```
Usunięto  7  wyników, gdzie wiek większy niż 100
```

Możemy sprawdzić czy historia kredytowa oraz czas pracy nie jest dłuższy niż długość życia:

```
Nie ma obserwacji, których czas pracy lub długość historii kredytowej jest większa niż długość życia
```

Dodatkowo sprawdzimy czy w danych nie występują duplikaty:

```
ilość duplikatów -  0
```

3. Analiza danych

Do podstawowej analizy danych możemy wykorzystać describe()

	person_age	person_income	person_emp_exp	loan_amnt	\
count	44993.000000	4.499300e+04	44993.000000	44993.000000	
mean	27.748428	7.990845e+04	5.394528	9583.176761	
std	5.909737	6.332213e+04	5.927159	6314.802655	
min	20.000000	8.000000e+03	0.000000	500.000000	
25%	24.000000	4.719500e+04	1.000000	5000.000000	
50%	26.000000	6.704600e+04	4.000000	8000.000000	
75%	30.000000	9.577800e+04	8.000000	12237.000000	
max	94.000000	2.448661e+06	76.000000	35000.000000	

	loan_int_rate	loan_percent_income	cb_person_cred_hist_length	\
count	44993.000000	44993.000000	44993.000000	
mean	11.006448	0.139736	5.866557	
std	2.978985	0.087207	3.877167	
min	5.420000	0.000000	2.000000	
25%	8.590000	0.070000	3.000000	
50%	11.010000	0.120000	4.000000	
75%	12.990000	0.190000	8.000000	
max	20.000000	0.660000	30.000000	

	credit_score	loan_status
count	44993.000000	44993.000000
mean	632.585713	0.222257
std	50.402411	0.415767
min	390.000000	0.000000
25%	601.000000	0.000000
50%	640.000000	0.000000
75%	670.000000	0.000000
max	784.000000	1.000000

Jak mogliśmy się domyślić młodzi ludzie najczęściej biorą pożyczki.

Sprawdźmy ilu procentom osób został udzielony kredyt

person_gender	female	male
loan_status		
0	0.777497	0.777943
1	0.222503	0.222057

Tylko 22% udzielono kredytu. W naszych danych dominują te obserwacje w których nie udzielono pożyczki.

4. Wybranie istotnych zmiennych

Analizę zmiennych zaczniemy od zmiennych kategorycznych

person_gender	female	male
loan_status		
0	0.777497	0.777943
1	0.222503	0.222057

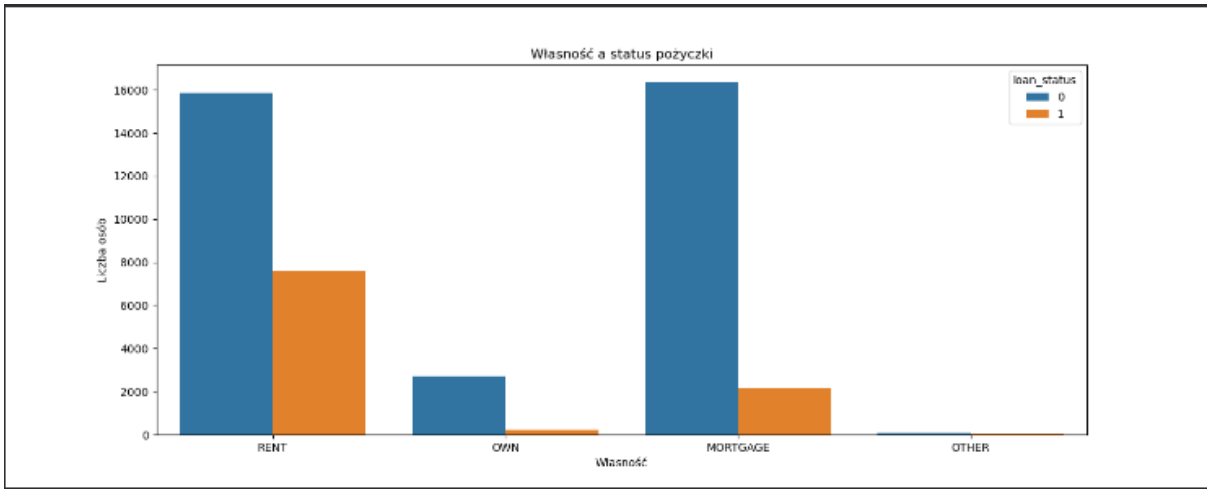
Dla płci można wywnioskować, że nie odgrywa ona roli w dostaniu pożyczki. Jest to dosyć intuicyjne.

person_education	Associate	Bachelor	Doctorate	High School	Master
loan_status					
0	0.779644	0.774709	0.771337	0.776859	0.782378
1	0.220356	0.225291	0.228663	0.223141	0.217622

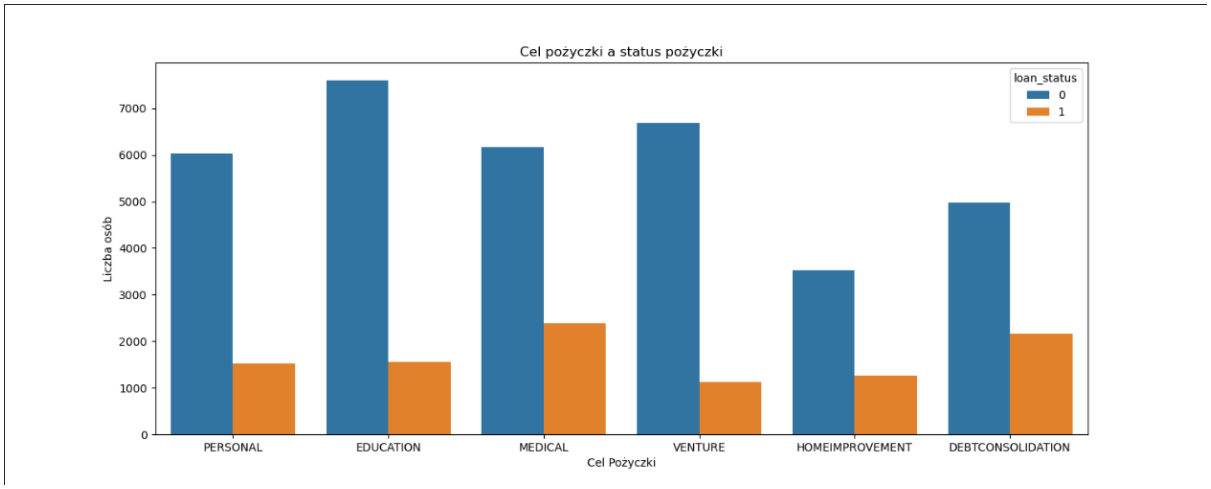
Dla edukacji można również wywnioskować, że nie odgrywa ona znaczącej roli do uzyskania pożyczki

loan_status		
0	0.548268	1.0
1	0.451732	0.0

Ta zmienna jest najbardziej znacząca z powyższych, gdyż każda osoba mająca jakiekolwiek problemy ze spłatą wcześniejszej pożyczki nie dostanie pożyczki



Ta zmienna może być ważna, ponieważ ma różne proporcje w zależności od własności



Cel pożyczki, także decyduje o tym czy jest on udzielany

Z naszych danych możemy wywnioskować ciekawą zalezność:

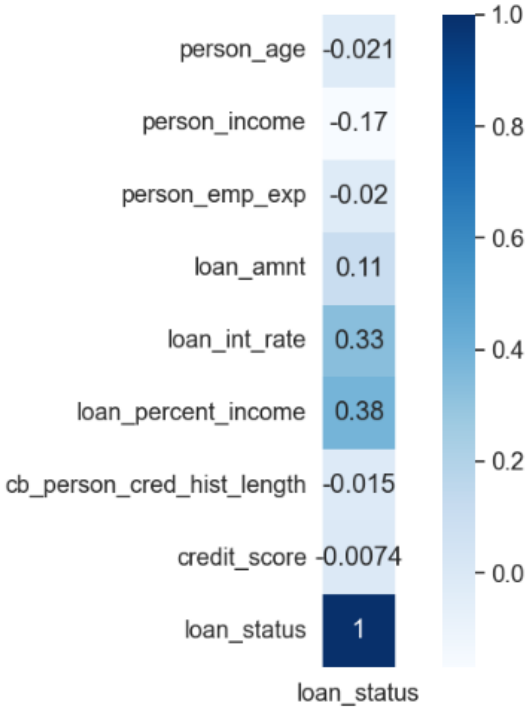
$$\frac{loan\_amnt}{Person\_income} * 100\% = loan\_percent\_income$$

Korelacja tych zmiennych wynosi:

```
0.999445497695405
```

Odrzucamy zmienną loan_percent_income, ponieważ wartość korealacji wynosi prawie 1

Przyjrzyjmy się korelacji między zmiennymi liczbowymi a lean_status



Możemy odrzuć zmienne, których wartość jest bliska zeru

5. Przygotowanie danych

Dzielimy dane na predyktory oraz zmienną celu. Następnie dane dzielimy na zbiór uczący i zbiór testowy, gdzie 30% danych to testowe.

6. Sieci neuronowe

Tworzymy pipeline, który jako pierwszy etap ma odpowiednią zamianę predyktorów kategorycznych oraz numerycznych. W przypadku zmiennych numerycznych stosuje normalizacje do przedziału [0,1] - ten przedział został wybrany dzięki grid search. W przypadku zmiennych kategorycznych używamy tak zwanego oneHotEncodera, to znaczy, że dla każdej wartości z danej zmiennej kategorycznej tworzymy osobną kolumnę. Następnym etapem jest wykonanie MLPClassifier z następującymi hiperparametrami:

random_state – ziarno generatora, pozwala na uzyskanie powtarzalności wyników. W naszym przypadku jest to numer indeksu.

max_iter – maksymalna ilość epok. W naszym przypadku jest to 1000, gdyż ilość danych jest dosyć duża.

hidden_layer_sizes = *ilość nerunów w ukrytej warstwie. W naszym przypadku (50, 50), co oznacza 2 ukryte warstwy z 50 neuronami każda*.

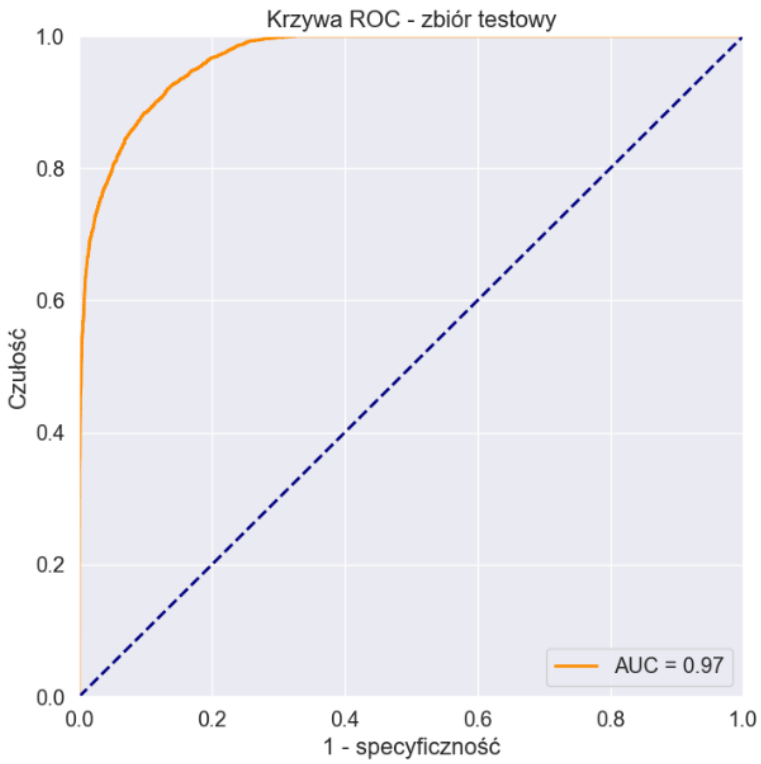
Te hiperparmetry zostały znalezione, dzięki grid search.

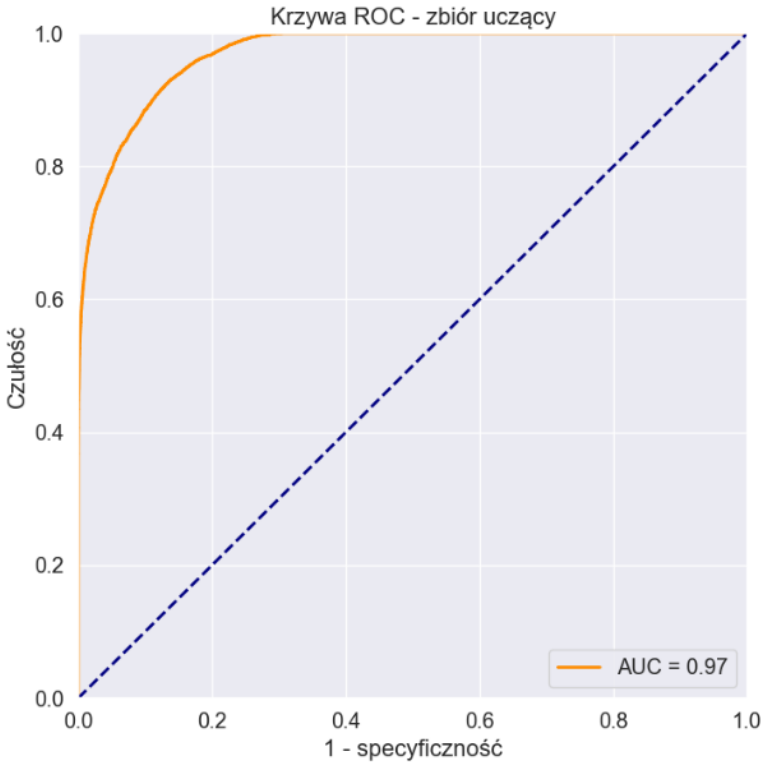
Następnie uzyskujemy następujące wyniki:

```
Macierze pomyłek
Zbiór uczący:
       0     1
0  10109   330
1    758  2301
Zbiór testowy:
       0     1
0  23776   778
1   1731  5210
```

```
Trafność:
Zbiór uczący:  0.9203365613589458
Zbiór testowy:  0.9193954659949622
```

```
Czułość:
Zbiór uczący:  0.7506123037026365
Zbiór testowy:  0.7522066034651848
Swoistość:
Zbiór uczący:  0.9683147348700822
Zbiór testowy:  0.9683877766069547
Precyzja:
Zbiór uczący:  0.8700734802939212
Zbiór testowy:  0.8745724059293044
F1:
Zbiór uczący:  0.8059401345811741
Zbiór testowy:  0.8087873462214411
```





7. XGBOOST

Tworzymy pipeline, który jako pierwszy etap ma odpowiednią zamianę predyktorów kategorycznych. W przypadku zmiennych numerycznych zostawiamy wartości takie jakie one mają. W przypadku zmiennych kategorycznych używamy tak zwanego oneHotEncodera, to znaczy, że dla każdej wartości z danej zmiennej kategorycznej tworzymy osobną kolumnę. Następnym etapem jest wykonanie XGBClassifier z następującymi hiperparametrami:

objective *– ustawiamy czy to regresja czy klasyfikacja, w naszym przypadku klasyfikacja binarna*

random_state – ziarno generatora, pozwala na uzyskanie powtarzalności wyników. W naszym przypadku jest to numer indeksu.

max_depth – maksymalna głebokość drzewa. W naszym przypadku jest to 7.

n_estimators - ilość drzew. W naszym przypadku 500.

subsample – procent danych wykorzystanych do budowy drzew. W naszym przypadku jest to 80%.

learning_rate – tempo uczenia. W naszym przypadku 0.1

gamma *- wartość, która im jest większa tym bardziej hamuje podział drzew. W naszym przypadku jest to 1.*

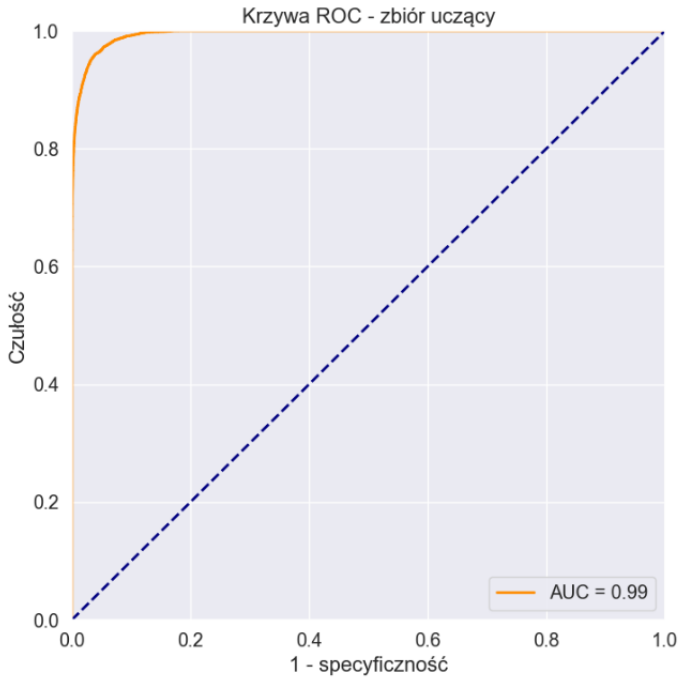
Te hiperparmetry zostały znalezione, dzięki grid search.

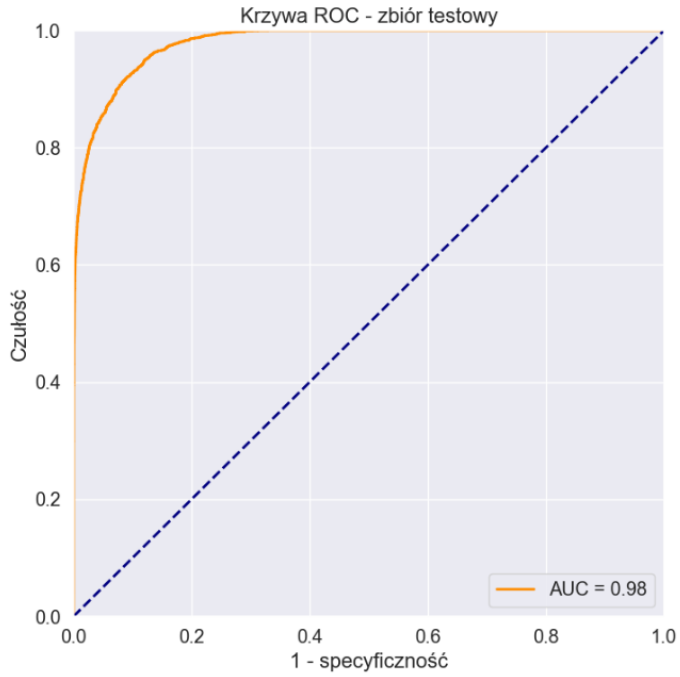
Następnie uzyskujemy następujące wyniki:

```
Macierze pomyłek:
Zbiór uczący:
       0     1
0  24261   293
1    811  6130
Zbiór testowy:
       0     1
0  10162   277
1    612  2447
```

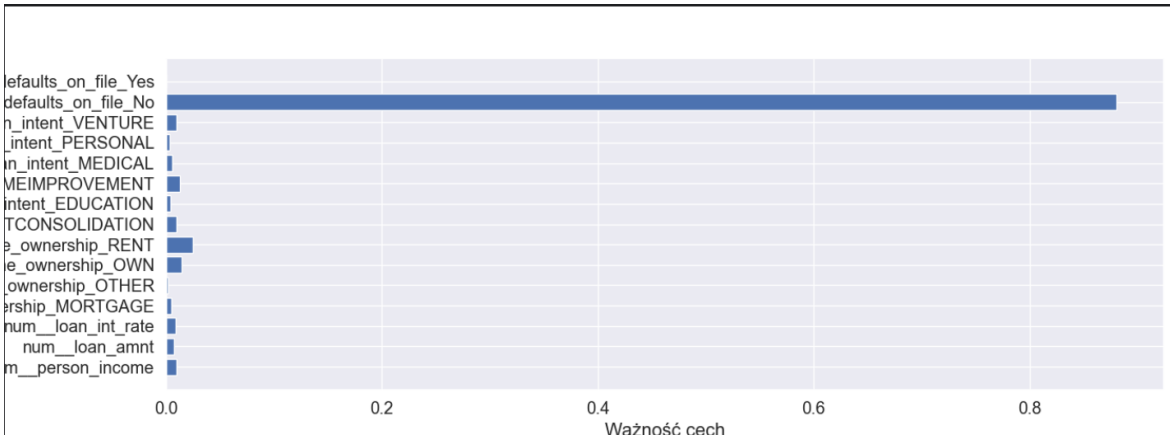
```
Trafność:
Zbiór uczący:  0.9649468169550722
Zbiór testowy:  0.9341383908727219
```

```
Czułość:
Zbiór uczący:  0.7999346191565871
Zbiór testowy:  0.8831580463910099
Swoistość:
Zbiór uczący:  0.9734648912731104
Zbiór testowy:  0.9880671173739513
Precyzja:
Zbiór uczący:  0.8983113069016153
Zbiór testowy:  0.9543826872178109
F1:
Zbiór uczący:  0.84627356043576
Zbiór testowy:  0.9173900029931157
```





Dodatkowo możemy sprawdzić, które zmienne miały istotny wpływ na wynik



Jak możemy zauważyć, sprawdza się to z tym co zakładaliśmy wcześniej, że zmienna defaults_on_file jest najistotniejsza.

8. Wnioski

Jak możemy zauważyć, trafność obu modeli wynosi ponad 90%, co jest dobrym wynikiem. W przypadku sieci neuronowych przeuczenie jest mniejsze niż w przypadku XGboosta, jednak jest ono w granicach normy. Wynik dla XGboosta jest odrobinę lepszy niż dla sieci neuronowych. Możemy także zauważyć, że swoistość w obu modelach jest większa od czułości. Jak wiadomo, jeżeli jeden z tych wskaźników rośnie to drugi maleje. Wtedy musimy się zastanowić na czym nam bardziej zależy. W naszym przypadku lepszym podejściem jest większa swoistość, gdyż lepszym podejściem jest unikanie ryzyka przyznania pożyczki osobom, które mogą jej nie spłacić.