

Przewidywanie choroby wątroby

Maciej Kapitan

maciej.kapitan@student.pk.edu.pl

Urszula Wąsowska

urszula.wasowska@student.pk.edu.pl

11 czerwca 2023

1 Abstract

Projekt zawiera opis informatycznych metod klasyfikacji guzów piersi na złośliwe i niezłośliwe, na podstawie danych z badań medycznych. Dokument zawiera opis zbioru danych, w którym jest on badany metodami statystycznymi. Następnie analizowana jest literatura na ten temat, motywacja do stworzenia projektu i ewaluacja. Kolejnym punktem jest opis zastosowanych zasobów oraz metod informatycznych potrzebnych do stworzenia projektu. Na końcu znajduje się opis eksperymentu w którym można wyróżnić trzy etapy:

- preprocessing danych
- trening modelu
- optymalizacja

2 Wprowadzenie

Machine learning jest szybko rozwijającą się dziedziną, która znajduje wiele zastosowań w różnych dziedzinach, w tym w medycynie. Program komputerowy może wyremontować lekarzy w rozpoznawaniu chorób na podstawie wyników badań. W naszej pracy będziemy zajmowali się badaniem guzów piersi - nasz program pomoże ocenić czy dany nowotwór jest złośliwy czy nie.

3 Zbiór danych

Zbiór danych pochodzi ze strony kaggle.com i zawiera dane pacjentów których przebadano pod kątem złośliwości guza piersi. Zbiór zawiera następujące dane z badań:

link do zbioru: <https://www.kaggle.com/datasets/yassserh/breast-cancer-dataset>

- radius mean - Średnia odległość od centrum do punktów na obwodzie guza.
- texture mean: Odchylenie standardowe wartości skali szarości pikseli w guzie.
- perimeter mean: Obwód guza.
- area mean: Powierzchnia guza.
- smoothness mean: Lokalna zmienność długości promieni guza.
- compactness mean: Obwód do kwadratu dzielony przez powierzchnię - 1.0.
- concavity mean: Stopień wklęsłości fragmentów konturu.
- concave points mean: Liczba wklęsłych fragmentów konturu.
- symmetry mean: Symetria guza.
- fractal dimension mean: "Przybliżenie linii brzegowej" - 1.
- radius se: Błąd standardowy odległości od centrum do punktów na obwodzie.
- texture se: Błąd standardowy wartości skali szarości pikseli.
- perimeter se: Błąd standardowy obwodu guza.
- area se: Błąd standardowy powierzchni guza.
- smoothness se: Błąd standardowy lokalnej zmienności długości promieni.
- compactness se: Błąd standardowy Obwód do kwadratu dzielony przez powierzchnię - 1.0.
- concavity se: Błąd standardowy stopnia wklęsłości fragmentów konturu.
- concave points se: Błąd standardowy liczby wklęsłych fragmentów konturu.
- symmetry se: Błąd standardowy symetrii guza.
- fractal dimension se: Błąd standardowy "przybliżenia linii brzegowej"
- radius worst: Największa wartość średnia odległość od centrum do punktów na obwodzie.
- texture worst: Największa wartość odchylenia standardowego wartości skali szarości pikseli.
- perimeter worst: Największa wartość obwodu guza.
- area worst: Największa wartość powierzchni guza.

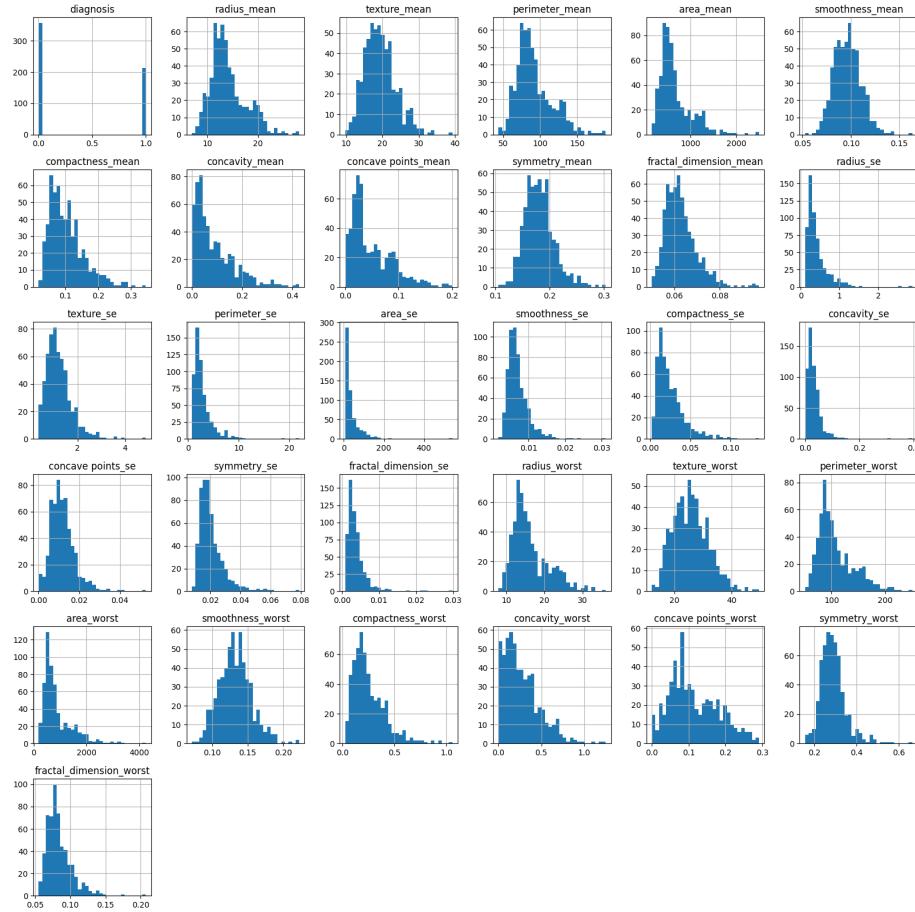
- smoothness worst: Największa wartość lokalnej zmienności długości promieni.
- compactness worst: Największa wartość - Obwód do kwadratu dzielony przez powierzchnię - 1.0.
- concavity worst: Największa wartość stopnia wklęsłości fragmentów konturu.
- concave points worst: Największa wartość liczby wklęsłych fragmentów konturu.
- symmetry worst: Największa wartość symetrii guza.
- fractal dimension worst: Największa wartość "przybliżenia linii brzegowej"

Dane wyglądają następująco w tabelce:

	Age of the patient	Is male	Total Bilirubin	Direct Bilirubin	Alkaline Phosphatase	Sgpt Alanine Aminotransferase	Sgot Aspartate Aminotransferase	Total Proteins	ALB Albumin	A/G Ratio	Albumin and Globulin Ratio	Result
0	65.0	0.0	0.7	0.1	187.0	16.0	18.0	6.8	3.3	0.90	0	
1	62.0	1.0	10.9	5.5	699.0	64.0	100.0	7.5	3.2	0.74	0	
2	62.0	1.0	7.3	4.1	490.0	60.0	68.0	7.0	3.3	0.89	0	
3	58.0	1.0	1.0	0.4	182.0	14.0	20.0	6.8	3.4	1.00	0	
4	72.0	1.0	3.9	2.0	195.0	27.0	59.0	7.3	2.4	0.40	0	
5	46.0	1.0	1.8	0.7	208.0	19.0	14.0	7.6	4.4	1.30	0	
7	29.0	0.0	0.9	0.3	202.0	14.0	11.0	6.7	3.6	1.10	0	
8	17.0	1.0	0.9	0.3	202.0	22.0	19.0	7.4	4.1	1.20	1	
9	55.0	1.0	0.7	0.2	290.0	53.0	58.0	6.8	3.4	1.00	0	
10	57.0	1.0	0.6	0.1	210.0	51.0	59.0	5.9	2.7	0.80	0	

3.1 Histogramy

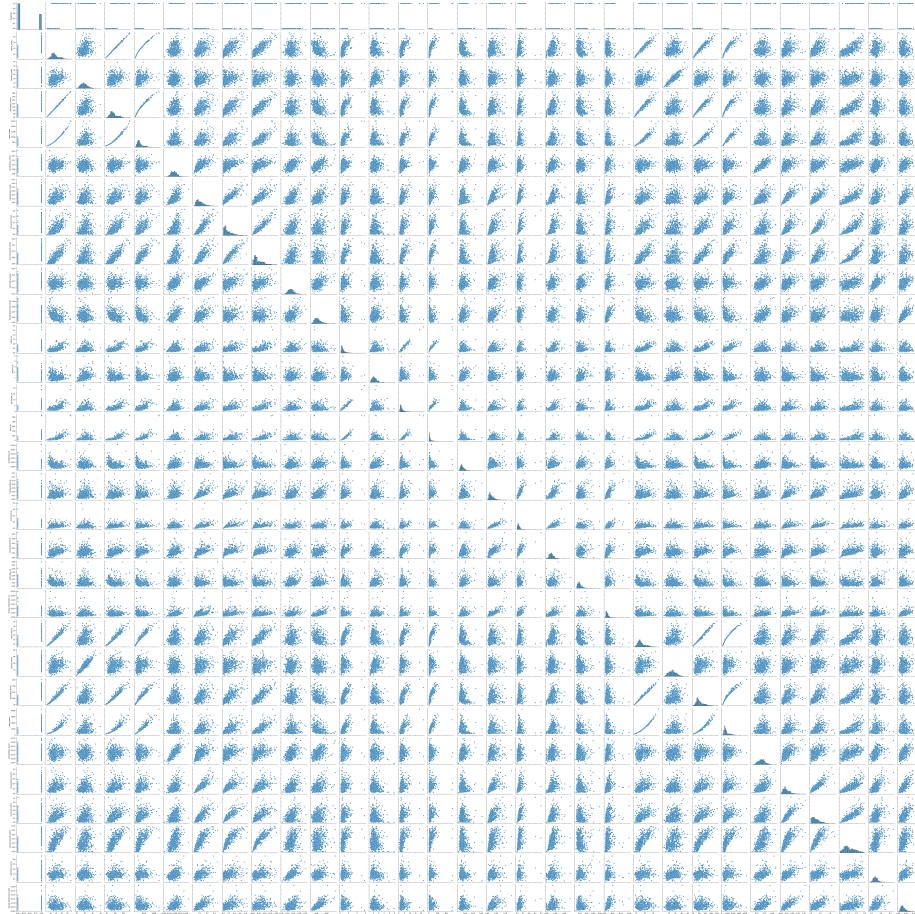
Histogramy danych wyglądają następująco:



Analizując histogramy poszczególnych kolumn można zauważyć że wiele z nich ma rozkłady przypominające rozkład normalny (czasami z większą lub mniejszą skośnością). Jest to dobry znak, ponieważ regresja i klasyfikacja dają zazwyczaj lepsze wyniki dla danych z rozkładem normalnym. Niestety można też dostrzec wartości odstającym które niestety pogarszają końcowy efekt - często wartości odstające mają większy wpływ na wynik regresji i klasyfikacji co może nieco zniekształcić model.

W kolumnie z wynikami widać przewagę klasy 0

3.2 Pairplot



Analizując pairplot można porównać zależności pomiędzy danymi. Ponieważ zbiór ma aż 31 kolumn, nie będziemy analizować wszystkich korelacji.

Pairploty zmiennych niezależnych ze zmienną zależną - na wielu wykresach można zauważyc, że dla różnych wartości zmiennej zależnej (diagnosis) rozkłady zmiennych niezależnych różnią się od siebie. Np dla zmiennej radius_mean - przedział wartości w których diagnosis=0 jest węższy i bardziej przesunięty w lewo niż przedział wartości dla których diagnosis=1. To oznacza, że po dużych wartościach promienia guza, możemy orzec, że jest on złośliwy.

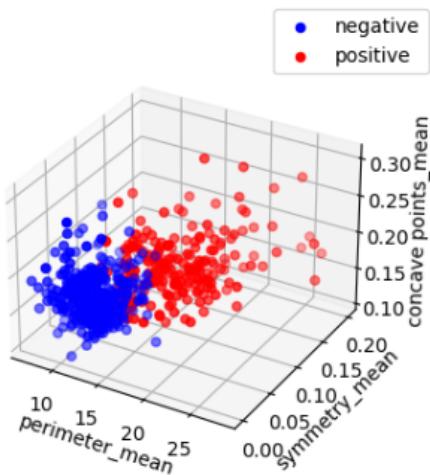
Pairploty dwóch zmiennych niezależnych - na niektórych wykresach np (radius_mean i perimeter_mean) można zauważyc silną zależność liniową pomiędzy tymi danymi. Jest to w miarę logiczne, ponieważ radius_mean oznacza promień guza natomiast perimeter_mean jego obwód, a jak wiadomo te dwie dane są liniowo powiązane. Jeśli mamy dwie dane niezależne, silnie powiązane liniowo, możemy wyeliminować jedną z nich z modelu regresji/klasyfikacji, ponieważ jest ona nadmiarowa - skoro dana Y zależy od danej X liniowo, a Z

zależy do X i Y liniowo, to tak naprawdę Z zależy tylko do X.

Na wielu pairplotach dane są rozmiieszczono nieliniowo co sugeruje brak jakiegokolwiek zależności między danymi. Niestety na niektórych wykresach widać obecność obserwacji odstających - np w radius_mean i radius_se. To z kolei może powodować że regresja/klasyfikacja da nie do końca poprawne wyniki.

3.3 Scatterplot

Przygotowaliśmy scatter plot prezentujący rozmieszczenie różnych klas danych w zależności od trzech wybranych cech - kierowaliśmy się przy wyborze korelacją tych cech ze zmienną zależną. Na wykresie widać granicę między jedną klasą i drugą - to jest dobry znak z punktu widzenia klasyfikacji - można stwierdzić że łatwo będzie sklasyfikować dane



4 Przegląd literatury

- Mangasarian OL, Street WN, Wolberg WH. Breast cancer diagnosis and prognosis via linear programming. *Operations Research*. 1995;43(4):570-577. - W artykule "Breast cancer diagnosis and prognosis via linear programming" autorstwa Mangasariana, Streeta i Wolberga, opisane są badania nad zastosowaniem programowania liniowego do diagnozowania i prognozowania raka piersi. W artykule autorzy skupiają się na wykorzystaniu technik programowania liniowego do analizy danych medycznych dotyczących raka piersi. W szczególności, opracowali model matematyczny, który integruje różne zmienne diagnostyczne, takie jak wielkość guza, typ komórek nowotworowych, stopień zaawansowania, a także dane kliniczne pacjenta, w celu przewidywania diagnozy oraz prognozowania przeżycia.

- Wolberg WH, Street WN, Mangasarian OL. Machine learning techniques to diagnose breast cancer from fine-needle aspirates. *Cancer Letters*. 1994;77(2-3):163-171. W artykule "Machine learning techniques to diagnose breast cancer from fine-needle aspirates" autorstwa Wolberga, Streeta i Mangasariana, omawiane są techniki uczenia maszynowego do diagnostowania raka piersi na podstawie cienkoigłowych aspiratów. Artykuł ten został opublikowany w czasopiśmie Cancer Letters w 1994 roku. Autorzy skupiają się na wykorzystaniu technik uczenia maszynowego do analizy próbek cienkoigłowych aspiratów pobranych od pacjentek podejrzanych o raka piersi.
- Elter M, Schulz-Wendtland R, Wittenberg T, et al. Computer-aided diagnosis of breast cancer: the role of benign breast tissue heterogeneity in automated classification. *Academic Radiology*. 2005;12(4):487-495. Artykuł omawia zastosowanie komputerowego wspomagania diagnozy w przypadku raka piersi. Autorzy skupiają się na roli zróżnicowania tkanki piersiowej w automatycznej klasyfikacji nowotworów.
- Madabhushi A, Lee G. Image analysis and machine learning in digital pathology: Challenges and opportunities. *Medical Image Analysis*. 2016;33:170-175. Artykuł omawia wyzwania i możliwości związane z analizą obrazów i uczeniem maszynowym w cyfrowej patologii. Autorzy zwracają uwagę na rosnące znaczenie tych technik w diagnozowaniu i leczeniu chorób.
- Cruz-Roa A, Basavanhally A, González F, et al. Automatic detection of invasive ductal carcinoma in whole slide images with convolutional neural networks. In: Proceedings of the IEEE International Symposium on Biomedical Imaging; 2014.- Artykuł opisuje automatyczne wykrywanie naciekającego raka przewodowego w obrazach całych preparatów histologicznych za pomocą konwolucyjnych sieci neuronowych. Autorzy prezentują zastosowanie tych sieci w wykrywaniu i diagnozowaniu raka piersi na podstawie obrazów histologicznych.

5 Motivation

Dużą część pracy lekarzy stanowi diagnoza chorób na podstawie badań medycznych. Wymaga to analizy dużej ilości danych. Czasami aby stwierdzić występowanie danej choroby należy przeanalizować wiele parametrów i połączyć je w jedną całość żeby zweryfikować występowanie danej choroby. Z pomocą mogą przyjść metody uczenia maszynowego. Na podstawie danych tabelarycznych z badań medycznych, wytrenowany model może oceniać czy stan pacjenta jest prawidłowy. Przeniesienie obowiązku diagnozy z człowieka na komputer może zaoszczędzić wiele czasu i pieniędzy oraz wyeliminować błędy naturalnie popełniane przez ludzi - z pomocą systemu komputerowego maleje ryzyko że lekarz błędnie oceni stan pacjenta lub pominie niektóre ważne dane

6 Evaluation

W analizie danych medycznych niezwykle ważna jest precyza, ponieważ stawką jest ludzkie życie. System musi mieć jak największą precyzję w przewidywaniu danej choroby. Szczególnie ważne jest aby wynik programu dawał jak najmniejszą ilość przypadków false negative, czyli takich które dla pozytywnego przypadku dają negatywny wynik. Jeśli pacjent ma chorobę, a system uzna że jest on zdrowy, to na podstawie tego lekarz może nie skierować go na dalsze badanie lub leczenie co będzie powodować dalszy rozwój choroby. Od naszego modelu oczekujemy aby zwracał możliwie jak najmniejszą ilość takich przypadków, nawet kosztem przypadków false positive, czyli takich gdzie system stwierdza chorobę podczas gdy pacjent jest zdrowy. Wtedy lekarz może ponownie przeanalizować wyniki i jeśli nie stwierdzi nieprawidłowości - odesłać pacjenta do domu.

7 Zasoby

Projekt będzie zaimplementowany w języku python z pomocą notatnika jupyter oraz pakietów służących do analizy danych: numpy, pandas, sklearn, seaborn, itd. Dane natomiast będą odczytywane z pliku csv.

8 Zastosowane metody

W naszym projekcie skorzystaliśmy z klasyfikatorów implementowanych w bibliotece scikit-learn w języku python. Krótko omówimy kilka z nich:

8.1 Klasyfikator VotingTeamClassification

Jest to algorytm uczenia maszynowego, który łączy wyniki kilku różnych klasyfikatorów w celu podjęcia ostatecznej decyzji. Klasyfikatory, które są wykorzystywane w ramach tego algorytmu, mogą być różnych typów, takich jak drzewa decyzyjne, sieci neuronowe, maszyny wektorów nośnych itp.

8.2 Klasyfikator StackingTeamClassification

Jest to algorytm uczenia maszynowego, który łączy wyniki kilku różnych klasyfikatorów za pomocą dodatkowego meta-klasyfikatora w celu podjęcia ostatecznej decyzji. Jest to technika stosowana w zespołowych metodach uczenia maszynowego, gdzie klasyfikatory bazowe są wykorzystywane jako "ekspertzy", a meta-klasyfikator uczy się, jak skutecznie połączyć ich wyniki.

8.3 XGBoostClassification

Jest to klasyfikator oparty na algorytmie gradient boosting, który jest bardzo popularny w dziedzinie uczenia maszynowego i analizy danych. XGBoost (eXtreme Gradient Boosting) jest implementacją gradient boosting na bazie drzew

decyzyjnych, która charakteryzuje się wysoką wydajnością i skutecznością w rozwiązywaniu problemów klasyfikacji. Klasyfikator XGBoostClassification jest ceniony ze względu na swoją zdolność do radzenia sobie z różnymi typami danych, obsługiwania zarówno cech numerycznych, jak i kategorycznych, a także dostosowywania się do różnych problemów klasyfikacji, takich jak binarna klasyfikacja, wieloklasowa klasyfikacja i klasyfikacja rankingowa. Dzięki swojej skuteczności i efektywności jest szeroko stosowany w praktyce analizy danych i konkursach uczenia maszynowego.

8.4 LGBMClassification

Jest to klasyfikator oparty na technologii gradient boosting, który używa metody LightGBM do rozwiązywania problemów klasyfikacji. LightGBM jest rozwinięciem algorytmu gradient boosting, które cechuje się wysoką wydajnością i możliwością obsługi dużych zbiorów danych.

8.5 Klasyfikatory wchodzące w skład voting classifier

8.5.1 Logistic Regression (Regresja Logistyczna)

Regresja logistyczna to metoda statystyczna stosowana do klasyfikacji binarnej. Wykorzystuje funkcję logistyczną do przewidywania prawdopodobieństwa przynależności do danej klasy.

8.5.2 Random Forest Classifier (Klasyfikator Lasu Losowego)

Las losowy to zbiór drzew decyzyjnych, gdzie każde drzewo jest trenowane na podstawie innego podzbioru danych treningowych. Ostateczna klasyfikacja jest dokonywana na podstawie głosowania większościowego drzew.

8.5.3 Klasyfikator SVC (Support Vector Classifier)

Jest to algorytm uczenia maszynowego wykorzystywany do problemów klasyfikacji. Opiera się na metodzie maszyny wektorów nośnych (SVM) i ma za zadanie znaleźć optymalną hiperpłaszczyznę separującą klasy w przestrzeni cech. Klasyfikator SVC używa funkcji jądra do obliczania podobieństwa między danymi wejściowymi a wzorcami treningowymi. Podczas treningu dobierane są optymalne hiperparametry, takie jak parametr regularyzacji C i parametr gamma (dla jądra RBF). Po treningu model SVC może klasyfikować nowe dane, mapując je na przestrzeń cech i przypisując do odpowiedniej klasy na podstawie położenia względem hiperpłaszczyzny. Klasyfikator SVC jest szczególnie skuteczny w przypadku danych o dużej liczbie cech lub gdy dane nie są liniowo separowalne w przestrzeni cech.

8.5.4 Klasyfikator GaussianNB (Gaussian Naive Bayes)

Algorytm ten oblicza parametry rozkładu normalnego dla każdej klasy i cechy wejściowej. Następnie, na podstawie tych parametrów, oblicza prawdopodobieństwo przynależności nowych danych do każdej klasy. Ostateczna klasyfikacja jest dokonywana na podstawie najwyższego prawdopodobieństwa przynależności do konkretnej klasy. Klasyfikator GaussianNB jest szybki i wydajny, szczególnie w przypadku danych ciągłych i dużej liczby cech.

9 Eksperyment

9.1 Preprocessing

- Skalowanie danych

Aby trening modelu był skuteczny dane muszą być odpowiednio przeskalone. Jeśli w kolumnie A dane mają szeroki zakres, natomiast w kolumnie B - wąski, system może interpretować dane z kolumny a jako "ważniejsze", ponieważ będą one miały większy wpływ na wynik. Do skalowania danych służą dwie podstawowe metody

- standart scaling

Standaryzacja to metoda, w której skaluje się cechy w taki sposób, aby średnia wartość była równa 0, a odchylenie standardowe 1

```
def standart_scaling(data):
    flat_data = data.flatten()
    mean = np.nanmean(flat_data)
    std = np.nanstd(flat_data)
    scaled_data = np.zeros(data.size)
    for i, x in enumerate(flat_data):
        scaled_data[i] = (x-mean)/std
    scaled_data=scaled_data.reshape(data.shape)
    return scaled_data
```

- min-max scaling

Skalowanie min-max polega na przeskalowaniu wartości z rozkładu według ustalonej dolnej i górnej granicy

```
def min_max_scaling(data):
    flat_data = data.flatten()
    x_min = min(flat_data)
    x_max = max(flat_data)
    new_max = 1
    new_min = 0
    scaled_data = np.zeros(data.size)
    for i, x in enumerate(flat_data):
        scaled_data[i] = ((x-x_min) / (x_max-x_min)) * (new_max - new_min) + new_min
    scaled_data=scaled_data.reshape(data.shape)
    return scaled_data
```

- Uzupełnianie danych brakujących

Czasami zdarza się że w zbiorze występują brakujące dane, które trzeba uzupełnić za pomocą metod statystycznych. Do uzupełniania brakujących danych użyliśmy trzech metod:

- Zastępowanie przez średnią

W tej metodzie dane brakujące są zastępowane przez średnią z pozostałych danych w zbiorze

Zapełnianie danych za pomocą średniej

```
column = M[:, 0]
mean = np.mean(column[~np.isnan(column)])
column = np.where(np.isnan(column), mean, column)
print('Wartości wypełnione: ', column[:size_to_remove])
print('Wartości rzeczywiste: ', df['Age of the patient'][:size_to_remove].to_numpy())
Wartości wypełnione: [44.10267044 44.10267044 44.10267044 ... 44.10267044 44.10267044
44.10267044]
Wartości rzeczywiste: [65. 62. 62. ... 46. 34. 34.]
```

- Regresja

W tej metodzie dane brakujące są zapełniane za pomocą regresji. Model jest uczyony na pozostałych danych, natomiast brakujące dane są zastępowane wartościami przewidywanymi przez wytrenowany model

Zapełnianie danych za pomocą regresji

```
: from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import MinMaxScaler

y = M[:, 0]
X = M[:, 1:]

scaler = MinMaxScaler()
scaler.fit(X)
X = scaler.transform(X)

X_test = X[size_to_remove:, :]
X_train = X[:size_to_remove, :]

y_test = df.to_numpy()[:size_to_remove, 0]
y_train = y[:size_to_remove]

reg=LinearRegression()
reg.fit(X_train,y_train)

column = reg.predict(X_test)

print('Wartości wypełnione: ', column)
print('Wartości rzeczywiste: ', y_test)
print('mean_squared_error : ', mean_squared_error(y_test, column))

Wartości wypełnione: [43.43736858 44.19230816 44.24747409 ... 43.42702432 44.46114541
44.29103371]
Wartości rzeczywiste: [65. 62. 62. ... 46. 34. 34.]
mean_squared_error : 257.7531384431067
```

regresja dala słaby wynik, ponieważ dane o wieku nie są liniowo zależne od pozostałych

- Losowanie z rozkładu

W tej metodzie dane brakujące są losowane z rozkładu normalnego. Ta metoda działa tylko gdy rozkład zmiennej jest zbliżony do rozkładu normalnego

Z histogramu kolumny 'Age of the patient' widać że rozkład tej zmiennej przypomina rozkład normalny, dlatego wartości brakujące zostaną zapelnione poprzez losowanie z rozkładu normalnego

```
column = M[:, 0]
median = np.median(column[~np.isnan(column)])
std = np.std(column[~np.isnan(column)])
count = len(column)
print('Mediania:', median)
print('Odchylenie standardowe:', std)
print('Liczebnośc:', count)

column = np.where(np.isnan(column), np.random.normal(median, std, count), column)
print('wartosci wypełnione:', column[:size_to_remove])
print('wartosci rzeczywiste:', df['Age of the patient'])[:size_to_remove].to_numpy()

Mediania: 45.0
Odchylenie standardowe: 15.965090479757686
Liczebnośc: 27158
Wartosci wypełnione: [ 43.79000112 49.26722538 58.35709673 ... 50.59743014 39.49760931
46.57302921]
Wartosci rzeczywiste: [65. 62. 62. ... 46. 34. 34.]
```

9.2 Trening modelu

Zanim przystąpi się do treningu ważny jest odpowiedni podział zbioru danych na dane uczące i testowe. W naszym kodzie, funkcja odpowiedzialna za tą część dzieli cały zbiór danych na ilość części jaką podamy w argumencie funkcji - każda taka część staje się zbiorem testowym, natomiast reszta pozostaje zbiorem uczącym - więc dla argumentu 5 otrzymamy 5 par zbiór uczący, zbiór testowy w których zbiór testowy będzie stanowił 1/5 ilości danych natomiast 4/5 to będzie zbiór treningowy. Taki podział ma na celu utworzenie zbiorów do tzw. cross validation - metodzie która umożliwia sprawdzenie wydajności algorytmu na wszystkich danych. Metoda ta również zajmuje się proporcjonalnym podziałem klas pozitwnej i negatywnej, tak aby nie doszło do sytuacji w której w zbiorze testowym znajdzie się tylko jedna klasa

Dzielenie zbioru na podzbiory treningowe i testowe

```
In [155]: def get_array_without_indexes(arr, indexes):
    mask = [i for i in range(arr.shape[0]) if i not in indexes]
    return arr[mask, :]

def split_data(data, nr=5, label_index=0):
    positives = data[data[:, 0]==1, :]
    negatives = data[data[:, 0]==0, :]
    np.random.shuffle(positives)
    np.random.shuffle(negatives)
    result = []
    for i in range(nr):
        low = lambda arr: int(i*arr.shape[0]/nr)
        up = lambda arr: int((i+1)*arr.shape[0]/nr)
        test_set = np.vstack((positives[low(positives):up(positives), :], negatives[low(negatives):up(negatives), :]))
        train_set = np.vstack((get_array_without_indexes(positives, range(low(positives), up(positives))), 
                               get_array_without_indexes(negatives, range(low(negatives), up(negatives)))))

        np.random.shuffle(test_set)
        result.append((train_set, test_set))
    return result

splitted_data = split_data(data)
for t in splitted_data:
    train_set=t[0]
    test_set=t[1]
    assert len(np.vstack((train_set, test_set))) == len(data)
    assert np.vstack((train_set, test_set)).all() in data
    assert data.all() in np.vstack((train_set, test_set))
```

Po przygotowaniu danych przystępujemy do ich treningu. Ponieważ zależy nam na możliwie jak największym wyeliminowaniu błędów false positive, ustawiamy parametr class weight dla jedynki na wysoką wartość - dzięki temu algorytm wie że klasa pozitwna jest ważniejsza niż negatywna i będzie brał ją bardziej pod uwagę w obliczeniach. Jendak w porównaniu modeli nie zmieniamy

tego parametru, ponieważ porównujemy skuteczność cross walidacji - poza tym nie wszystkie z klasyfikatorów mają możliwość ustawienia tego parametru, natomiast optymalizacją zajmiemy się w dalszej części. Różnież nie ustawiamy parametru C dla klasyfikatorów (czyli tzw regularyzacji), a więc wyniki w tabelce mogą być trochę gorsze niż w rzeczywistości. Jednak optymalizacja parametrów regresji jest przedmiotem dalszej części

Wyniki prezentują się w następującej tabelce:

	VotingClassifier		Stacking Classifier		XGBoost		LightGBM	
	train	test	train	test	train	test	train	test
Standart scaling	0.94	0.92	0.98	0.82	0.99	0.7	1	0.69
Min max scaling	0.89	0.78	0.99	0.58	0.99	0.66	1	0.6

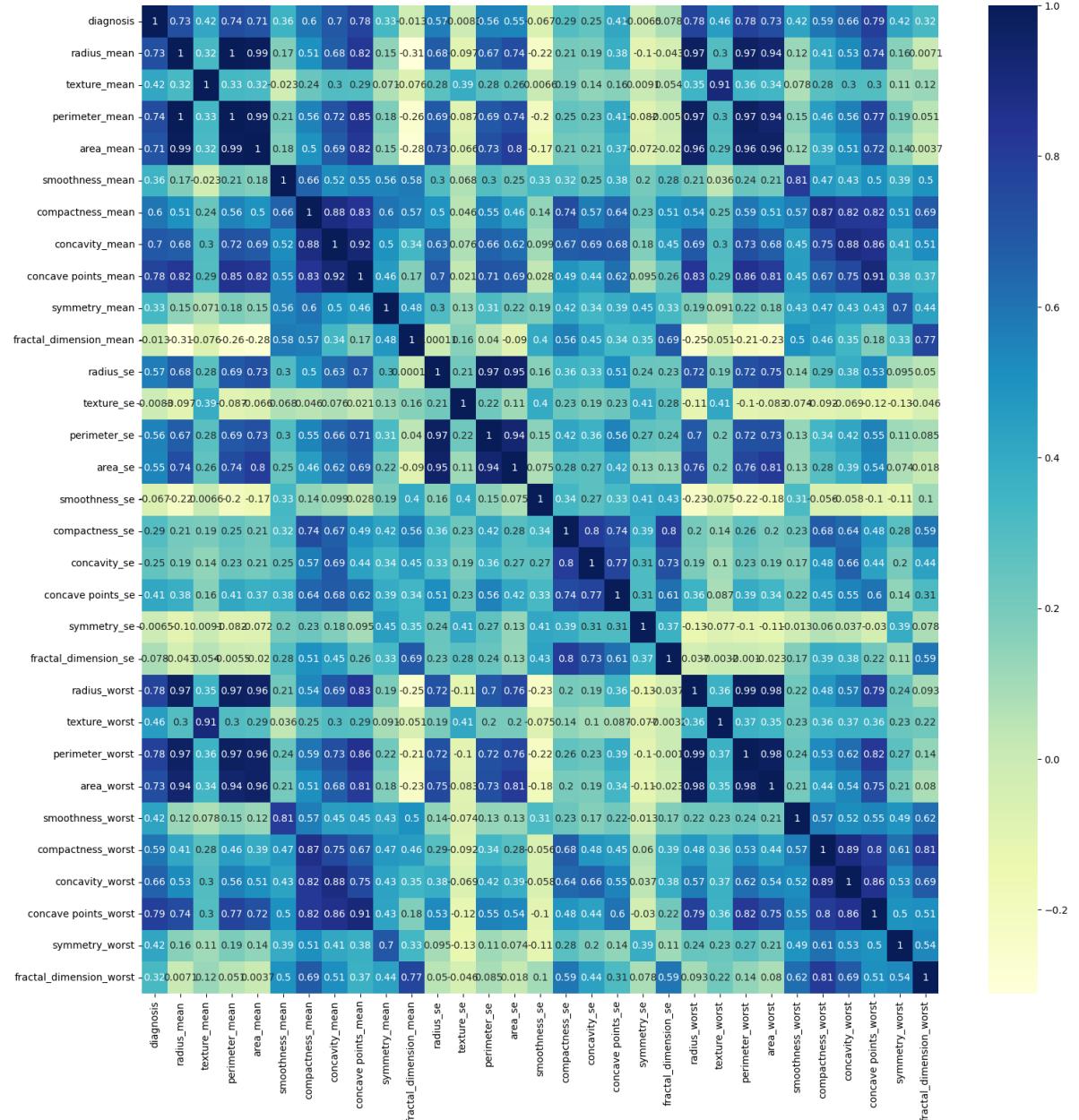
Analizując wyniki można zauważyc przewagę skalowania standaryzującego na skalowniem min-max - ta metoda uzyskała znacznie lepsze wyniki do drugiej. Po drugie najlepiej z klasyfikacją poradził sobie zdecydowanie VotingClassifier - ma skuteczność na zbiorze testowym znacznie przewyższającą inne klasyfikatory.

Najlepszą skutecznością treningową wykazały się z kolei klasyfikatory XGBoost i LightGBM - można jednak zauważyc dużą rozbieżność w efektywności na danych treningowych i testowych - jest to zjawisko przetrenowania (High Variance) które polega na zbyt dużym dopasowaniu się modelu do danych treningowych przez co słabo radzi sobie on z nowymi danymi. Szczegółowym omówieniem zajmieni się w ostatniej części rozdziału.

Dla naszych danych ta metoda uczenia zespołowego sprawdziła się najlepiej. W kolejnych rozdziałach będziemy zajmowali się ulepszeniem modelu z Voting Classifierem oraz omówieniem parametrów.

9.3 Optymalizacja

9.3.1 Pozbycie się zbędnych cech



Na podstawie macierzy korelacji odrzucamy te cechy które mają najbliższy zeru współczynnik korelacji ze zmienią zależną. Można również wyeliminować para-

metry które mają ze sobą bardzo wysokie współczynniki korelacji - korelacja na poziomie ok 99

Ze względu na małą korelację ze zmienną zależną możemy odrzucić parametry:

- fractal dimesion mean
- texture se
- smoothness se
- symmetry se
- fractal dimesion se

Ze względu na wzajemną korelację możemy odrzucić:

- perimeter mean (skorelowany z radius mean)
- area mean (skorelowany z radius mean)
- perimeter worst (skorelowany z radius worst)
- area worst (skorelowany z radius worst)

9.3.2 Optymalizacja parametrów z wykorzystaniem algorytmu genetycznego

Parametry do klasyfikatorów optymalizowaliśmy za pomocą algorytmu genetycznego. Ponieważ algorytmy genetyczne nie są przedmiotem analizy, nie będziemy omawiali szczegółowo ich działania. Funkcja której użyliśmy do oceny danego osobnika bierze ilość przypadków false negative i podnosi je do kwadratu i następnie dodaje ilość przypadków false positive - takie działanie ma na celu zwrócenie uwagi że błęd false negative jest z punktu widzenia wymagań modelu o wiele gorszym błędem niż false positive.

Parametry dla Random Forest Classifier:

- n_estimators=72
- min_samples_leaf=8
- max_depth=1
- min_samples_split=2
- max_features=0.6246773459100194
- class_weight=1: 31.333867295500973

Parametry dla Logistic Regression Classifier

- C=12.339539468229866
- max_iter=133

- solver=liblinear
- class_weight=1: 32.39031184034202

Parametry dla SVC Classifier

- C=90.30352499200572
- gamma=79.85352961598883
- kernel=linear
- coef0=0.840249635812267
- class_weight=1: 42.11248179061335

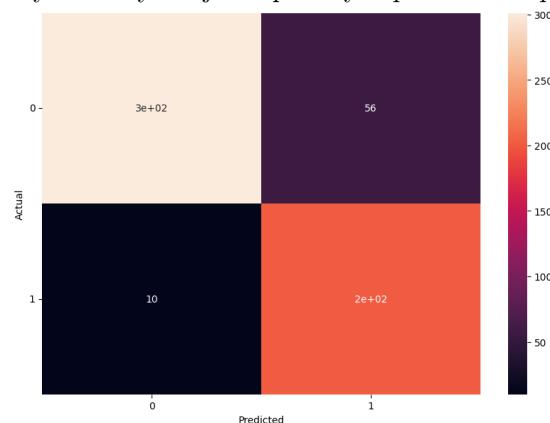
Parametry dla GaussianNB

- var_smoothing=0.4866792885754776

Można zauważyc że algorytm genetyczny dążył do zmaksymalizowania parametrów class weigth, ponieważ jak już wcześniej wyjaśniliśmy - dązymy do wyeliminowania błędów false negative

Kolejną rzeczą jest zwiększenie parametru C, który odpowiada za regularyzację. Zwiększyły współczynnik regularyzacji ma zapobiegać tzw. High Variation - czyli zjawisku przeuczenia. Powoduje ono że algorytm zbyt dobrze dopasowuje się do danych treningowych, przez co wypada gorzej dla testowych wartości. To powoduje dużą rozbieżność w efektywności dla zbioru treningowego i testowego (o wiele mniejsza skuteczność dla zbioru testowego). Aby temu zapobiec stosuje się metodę tzw regularyzacji, polegającej na dodawaniu do równania regresji/klasyfikacji odpowiednich wartości do równania.

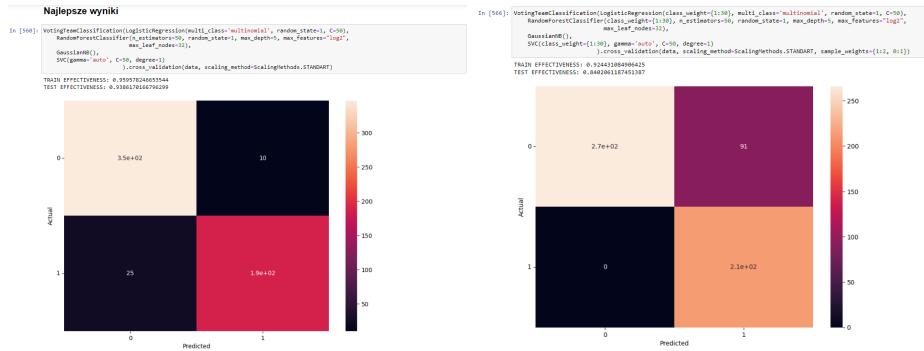
Wynik klasyfikacji dla podanych parametrów prezentuje się następująco:



Prawdopodobnie nie jest to najlepszy wynik klasyfikacji jaki można było otrzymać, jednak ze względu na duży czas obliczeń musieliszy znaczaco ograniczyć liczbę epok i osobników dla algorytmu genetycznego, ponieważ dla każdego osobnika, w każdej epoce liczona jest klasyfikacja, a sama w sobie zajmuje ona trochę czasu.

10 Podsumowanie

Podsumowując, wyniki klasyfikacji dały w miarę dobre rezultaty - uważamy że projekt może znaleźć zastosowanie w medycynie - pomoże to wzbogacić metody diagnostyki medycznej poprzez automatyczną weryfikację nowotworu na podstawie badań. Najlepszym klasyfikatorem okazał się Voting Classifier z odpowiednio dobranymi parametrami - w najlepszym wydaniu poprawnie klasyfikował nawet 94%



Metoda skalowania danych która okazała się najlepsza to skalowanie standaryzujące - sprawdziło się zdecydowanie lepiej niż skalowanie min-max. Można stwierdzić iż jest to spowodowane lekką anormalnością danych i dużą liczbą obserwacji odstających