P R O J E K T

Predykcja i analiza zbioru *Breast Cancer Diagnostic* przy użyciu algorytmów uczenia maszynowego (z ang. *Machine Learning*)

Anna Maria Ledwoń

Kraków 2020

Spis treści

1 Wstęp – Problem raka piersi

Rak piersi jest najczęściej występującym nowotworem złośliwym u kobiet w Polsce oraz jednym z największą ilością zgonów na świecie. Na raka piersi choruje jedna na osiem kobiet w krajach zachodnich. Co roku w Polsce diagnozuje się od 18 do 19 tys. przypadków tego nowotworu, z czego około 6 tys osób kończy się śmiertelnie. Wg opublikowanego w 2019 roku raportu Narodowego Instytutu Zdrowia – PZH w Warszawie w latach 2010-2016 współczynnik zgonów zwiększył się o 7,2%.

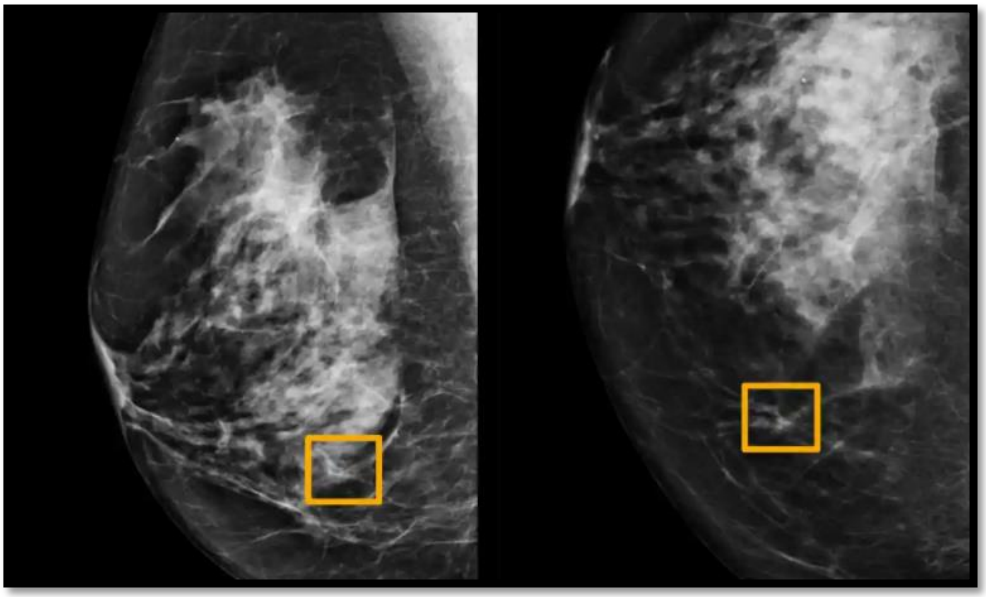
Badania naukowe dowodzą, że przyczyn choroby, jej przebieg i odpowiedź na leczenie są praktycznie identyczne u większości kobiet na całym świecie. Przeświadczenie, że rak piersi dotyka głównie kobiet białych w krajach uprzemysłowionych Ameryki Północnej i Europy Zachodniej jest nieprawdziwy, ponieważ w ostatnich latach obserwuje się gwałtowny skok liczby zachorowań w krajach Europy Wschodniej i Ameryki Łacińskiej. Aktualnie 70% wszystkich przypadków raka piersi na świecie dotyczy krajów rozwijających się [1].

Szansa wyleczenia raka piersi o średnicy 5 cm wynosi 50%, natomiast raka o średnicy 1 cm ponad 80%. Nic nie daje większej szansy na wyleczenie nowotwory jak wczesne wykrycie choroby. W krajach, w których są dostępne nowoczesne metody diagnostyczne to jak można się domyślać ok. 80-90% pacjentów przechodzi chorobę. W sytuacji gdy są one niedostępne, odsetek chorych, u których rak piersi rozpoznawany jest w stadium umożliwiającym wyleczenie, nie przekracza 50%. Skoro nadal w większości przypadków rak piersi prowadzi do zgonu, oznacza to, że więcej wysiłków należy położyć na wczesne wykrycie choroby [2].

Z tego powodu sztuczna inteligencja jest coraz częściej stosowana w diagnostyce, m.in. w rozpoznawaniu zdjęć rentgenowskich różnych schorzeń. Wykazano, że AI jest równie skuteczne w tej dziedzinie jak radiologowie w wykrywaniu raka piersi na zdjęciach rentgenowskich. Dodatkowo udowodniono, że zastosowanie nowych technologii zmniejsza odsetek błędu przy nieprawidłowej identyfikacji przez ludzkie oko lub całkowite przeoczenie guza na zdjęciu. Na ten moment wiadomo, że sztuczna inteligencja potrafi rozpoznawać zdjęcia rentgenowskie z podobną dokładnością jak doświadczeni radiologowie dlatego w przyszłości może pomóc personelowi medycznemu w diagnostyce trudnych do identyfikacji schorzeń lub bardzo małych guzów [3].

Naukowcy z Google Health, wykorzystali zbiór danych obejmujący zdjęcia z mammografii pacjentek z Wielkiej Brytanii i USA, u których zdiagnozowano różne postacie raka piersi. Na tej podstawie stworzyli model z wykorzystaniem narzędzi sztucznej inteligencji, który został wytrenowany na podstawie tych danych mammograficznych do wykrywania raka we wczesnych stadiach jego rozwoju.

Model AI po przejrzeniu dziesiątek tysięcy zdjęć był w stanie stwierdzić obecność raka piersi na podstawie najnowszej mammografii pacjentki na podobnym poziomie dokładności co bardzo doświadczeni radiologowie. Wykorzystanie algorytmów AI w diagnostyce nowotworów zmniejszyło błędy źle postawionej diagnozy w o 5,7% w przypadku danych z Wielkiej Brytanii i 1,2% dla przypadków z USA. Natomiast w przypadkach gdzie ludzkie oko mogło pominąć raka to model zdiagnozował aż o 9,4% więcej zdjęć nowotworów w Wielkiej Brytanii niż człowiek, a USA – o 2,7% [4].



Rys. 1. Żółte pole wskazuje, gdzie model AI wykrył raka ukrytego w tkance piersi. Sześciu wcześniejszym radiologom nie udało się wykryć raka w rutynowych badaniach mammograficznych [4].

Na powyższym przykładzie widać, że prowadzone badania z wykorzystaniem narzędzi sztucznej inteligencji dają nadzieję na lepszą diagnostykę i wczesne wykrycie raka piersi, a co za tym idzie większą szanse na wyleczenie. Z tego powodu w tej pracy dyplomowej postawiono sobie cel zbudowania modelu uczenia maszynowego na podstawie ogólnodostępnej bazy danych *Breast Cancer Wisconsin (Diagnostic) Data Set*, który będzie zdolny do odpowiedzi na pytanie *Czy pacjentka choruje na raka piersi złośliwego czy łagodnego?*.

2 Cel projektu

Celem projektu było:

- Podstawowa analiza zbioru *Breast Cancer Diagnostic*,
- Wizualizacja danych zbioru,
- Stworzenie modeli uczenia maszynowego,
- Predykcja z udziałem modeli Machine Learning,
- Porównanie modeli między sobą.

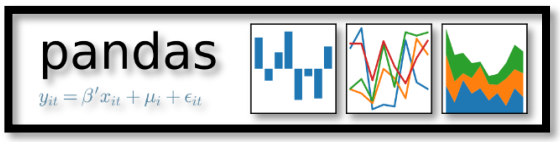
W realizacji projektu użyto Pythona jako jeżyka programowania oraz następujących bibliotek: Pandas, NumPy, Matplotlib, Seaborn, Scikit-learn i Keras. Natomiast zastosowanym środowiskiem programistycznym był Jupyter Notebook z pakietu Anaconda.

3 Narzędzie wykorzystane w projekcie

W tym rozdziale omówiono pakiety/biblioteki z jakich korzystano w celu analizy i predykcji zbioru *Breast Cancer Wisconsin (Diagnostic) Data Set*.

3.1 Pandas

Pierwszym pakietem, z którego skorzystano było *Pandas*. Jest to otwarto źródłowa biblioteka, zapewniająca wydajne i łatwe w użyciu struktury danych oraz narzędzia do analizy danych dla języka programowania Python.



Rys. 2. Logo pakietu Pandas.

Podstawowe funkcje/cechy/narzędzia biblioteki Pandas to:

- Obiekt DataFrame do manipulacji danymi ze zintegrowanym indeksowaniem.
- Narzędzia do odczytu i zapisu danych między strukturami danych w pamięci i różnymi formatami plików.
- Dopasowanie danych i zintegrowana obsługa brakujących danych.
- Przekształcanie i przestawianie zbiorów danych.
- Krojenie na podstawie etykiet, fantazyjne indeksowanie i podzbiór dużych zestawów danych.
- Wstawianie i usuwanie kolumny struktury danych.
- Grupuj według silnika umożliwiającego operacje podziel-zastosuj-połącz na zbiorach danych.
- Scalanie i łączenie zbiorów danych.
- Hierarchiczne indeksowanie osi do pracy z danymi wielowymiarowymi w strukturze danych o niższych wymiarach.
- Funkcjonalność szeregów czasowych: generowanie zakresu dat [4] i konwersja częstotliwości, statystyki ruchomego okna, regresje liniowe ruchomego okna, przesuwanie i opóźnianie dat.
- Zapewnia filtrację danych [5].

3.2 NumPy

Kolejnym pakietem jest *NumPy* to biblioteka języka programowania Python, dodająca obsługę dużych, wielowymiarowych tablic i macierzy, wraz z dużą kolekcją funkcji matematycznych wysokiego poziomu do obsługi tych tablic.

Rys. 3. Logo pakietu NumPy.

Używanie *NumPy* w Pythonie daje funkcjonalność porównywalną z MATLAB-em, ponieważ oba są interpretowane i oba pozwalają użytkownikowi pisać szybkie programy, o ile większość operacji działa na tablicach lub macierzach zamiast na skalarach. Dla porównania MATLAB oferuje dużą liczbę dodatkowych zestawów narzędzi, w szczególności Simulink, podczas gdy NumPy jest wewnętrznie zintegrowany z Pythonem, bardziej nowoczesnym i kompletnym językiem programowania [6].

3.3 Matplotlib

Matplotlib to biblioteka kreśląca dla języka programowania Python i jego rozszerzenia NumPy do matematyki numerycznej. Zapewnia zorientowany obiektowo interfejs API do osadzania wykresów w aplikacjach przy użyciu zestawów narzędzi GUI ogólnego przeznaczenia, takich jak Tkinter, wxPython, Qt lub GTK +. Istnieje również proceduralny interfejs „pylab” oparty na maszynie stanów (jak OpenGL), zaprojektowany tak, aby bardzo przypominał MATLAB, chociaż jego użycie jest odradzane [7].

Rys. 4. Logo pakietu Matplotlib.

3.4 Seaborn

Seaborn to biblioteka do wizualizacji danych w języku Python oparta na matplotlib. Zapewnia interfejs wysokiego poziomu do rysowania atrakcyjnych i pouczających grafik statystycznych [8].

Rys. 5. Logo pakietu Seaborn.

3.5 Scikit-learn

Scikit-learn to biblioteka do uczenia maszynowego wolnego oprogramowania dla języka programowania Python. Oferuje różne algorytmy klasyfikacji, regresji i grupowania, w tym maszyny wektorów pomocniczych, lasery losowe, wzmocnienie gradientu, k-średnie i DBSCAN, i jest zaprojektowany do współpracy z numerycznymi i naukowymi bibliotekami Pythona NumPy i SciPy [9].

Rys. 6. Logo pakietu Scrikit-learn.

3.6 Keras

Keras to biblioteka sieci neuronowych typu open source napisana w języku Python. Może działać na TensorFlow, Microsoft Cognitive Toolkit, R, Theano lub PlaidML. Zaprojektowany, aby umożliwić szybkie eksperymentowanie z głębokimi sieciami neuronowymi, koncentruje się na byciu przyjaznym dla użytkownika, modułowym i rozszerzalnym. Został on opracowany w ramach prac badawczych projektu ONEIROS (otwarty, neuro-elektroniczny inteligentny system operacyjny robota), a jego głównym autorem i opiekunem jest François Chollet, inżynier Google [10].

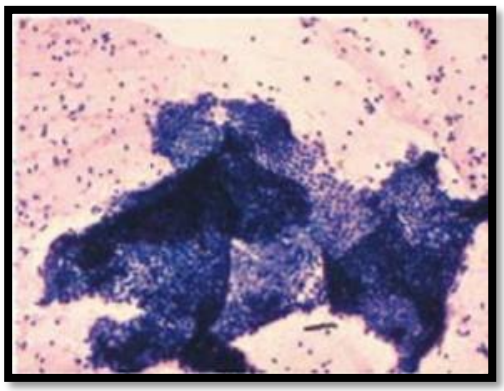
Rys. 7. Logo pakietu Keras.

4 Breast Cancer Wisconsin (Diagnostic) Data Set

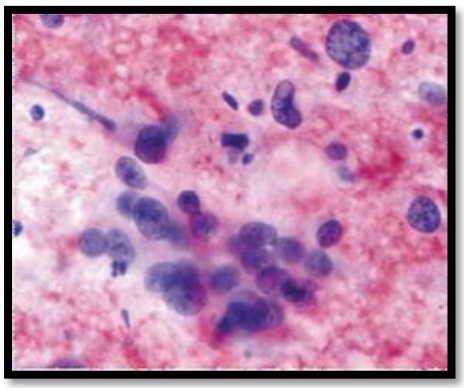
4.1 Omówienie bazy danych

Zbiorem danych, który poddano analizie był zbiór *Breast Cancer Wisconsin (Diagnostic) Data Set* z UCI Machine Learning Respository. Baza została opracowana na podstawie wyników badań histopatologicznych pacjentek, u których zdiagnozowano raka piersi złośliwego lub łagodnego pozyskanego przy pomocy biopsji aspiracyjnej cienkoigłowej (FNAB). Badanie daje informacje na temat charakterystyki komórek guza [11].

Na podstawie obrazów histopatologiczne guzów raka piersi pozyskane za pomocą metody FNAB badacze opisali 10 cech dla każdego raka złośliwego i łagodnego zebranego w zbiorze.



Rys. 8. Obraz cytologiczny gruczolaka-włókniaka śródprzewodowgo. Pow.150x. [12]



Rys. 9. Komórki raka przewodowego 3 stopnia wg Blomma-Richardsona. Pow. 200x [12].

Dodatkowo wszystkim pacjentkom nadano numer id i postawiono diagnozę, gdzie M w bazie danych oznacza raka złośliwego, a B – raka łagodnego. Każdemu przypadkowi nadano cechy opisane w poniższej tabeli. Pierwsze 10 zmiennych w zbiorze danych opisuje średnie wartości powyższych charakterystyk w próbce, drugie 10 zmiennych opisuje błąd standardowy, ostatnie 10 zmiennych opisuje „najgorsze" lub największe (średnią z trzech największych wartości pomiarów) wartości charakterystyk komórek w obrazie.

Tab. 1. Opisane cechy raka piersi zebrane
w *Breast Cancer Wisconsin (Diagnostic) Data Set*

L.p.	Nazwa cechy po polsku	Nazwa cechy po angielsku	Opis cechy
1.	Promień	**Radius**	Promień, czyli średnia odległość od środka do punktów brzegu
2.	Tekstura	**Texture**	Odchylenie standardowe wartości skali szarości
3.	Obwód	**Perimeter**	Obwód
4.	Obszar	**Area**	Pole zajmowane przez guza
5.	Gładkość	**Smoothness**	Lokalna zmienność długości promienia
6.	Zwartość	**Compactness**	Obwód^2/pole-1
7.	Wklęsłość	**Concavity**	Proporcja wklęsłych fragmentów brzegu
8.	Punkty wklęsłe	**Concave points**	Liczba wklęsłych fragmentów brzegu
9.	Symetria	**Symmetry**	Symetria
10.	Wymiar fraktalny	**Fractal dimension**	Wymiar fraktalny (przybliżenie linii brzegowej - 1)

Cały projekt został zrealizowany przy pomocy Pythona w środowisku programistycznym jakim jest Jupyter Notebook z pakietu Anaconda.

4.2 Import bazy danych

Na samym początku przed przystąpieniem do analizy. Zaimportowano wszystkie niezbędne i stosowane biblioteki w następujący sposób:

```
#import bibliotek
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.patches as mpatches
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.neighbors import KNeighborsClassifier
from keras.models import Sequential
from keras.layers import Dense, Activation
from keras.optimizers import SGD
from keras.utils.np_utils import to_categorical
from keras import callbacks
```

Następnie zaimportowano *Breast Cancer Wisconsin (Diagnostic) Data Set*, zmieniono nazwę kolum i wyświetlono informacji na temat zbioru.

```
#załadowanie zbioru danych
data = pd.read_csv('C:\PythonScripts\Python Thesis\wdbc.data')

# zmiana nazwy kolumny
data = data.rename(columns={'842302' : 'Id', 'M':'diagnosis',
'17.99': 'radius_mean', '10.38':'texture_mean', '122.8' :
'perimeter_mean', '1001': 'area_mean',
'0.1184':'smoothness_mean', '0.2776' : 'compactness_mean',
'0.3001':'concavity_mean', '0.1471': 'concave_points_mean',
'0.2419': 'symmetry_mean', '0.07871': 'fractal_dimension_mean',
'1.095' : 'radius_se', '0.9053':'texture_se', '8.589' :
'perimeter_se', '153.4' : 'area_se', '0.006399' :
'smoothness_se', '0.04904':'compactness_se', '0.05373' :
'concavity_se', '0.01587' : 'concave_points_se',
'0.03003':'symmetry_se', '0.006193':'fractal_dimension_se',
'25.38': 'radius_worst', '17.33':'texture_worst', '184.6':
'perimeter_worst', '2019': 'area_worst', '0.1622':
'smoothness_worst', '0.6656':'compactness_worst',
'0.7119':'concavity_worst', '0.2654': 'concave_points_worst',
'0.4601':'symmetry_worst', '0.1189': 'fractal_dimension_worst'})

# Wyświetlenie informacji o zbiorze
data.info()
```

Wynikiem polecenia data.info() jest tabela z nazwami kolumn, indeksem oraz typem zmiennych zebranych w danej kolumnie. Struktura bazy wygląda następująco:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 568 entries, 0 to 567
Data columns (total 32 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Id                       568 non-null    int64
 1   diagnosis                568 non-null    object
 2   radius_mean              568 non-null    float64
 3   texture_mean             568 non-null    float64
 4   perimeter_mean           568 non-null    float64
 5   area_mean                568 non-null    float64
 6   smoothness_mean          568 non-null    float64
 7   compactness_mean         568 non-null    float64
 8   concavity_mean           568 non-null    float64
 9   concave_points_mean      568 non-null    float64
 10  symmetry_mean            568 non-null    float64
 11  fractal_dimension_mean   568 non-null    float64
 12  radius_se                568 non-null    float64
 13  texture_se               568 non-null    float64
 14  perimeter_se             568 non-null    float64
 15  area_se                  568 non-null    float64
 16  smoothness_se            568 non-null    float64
 17  compactness_se           568 non-null    float64
 18  concavity_se             568 non-null    float64
 19  concave_points_se        568 non-null    float64
 20  symmetry_se              568 non-null    float64
 21  fractal_dimension_se     568 non-null    float64
 22  radius_worst             568 non-null    float64
 23  texture_worst            568 non-null    float64
 24  perimeter_worst          568 non-null    float64
 25  area_worst               568 non-null    float64
 26  smoothness_worst         568 non-null    float64
 27  compactness_worst        568 non-null    float64
 28  concavity_worst          568 non-null    float64
 29  concave_points_worst     568 non-null    float64
 30  symmetry_worst           568 non-null    float64
 31  fractal_dimension_worst  568 non-null    float64
dtypes: float64(30), int64(1), object(1)
memory usage: 142.1+ KB
```

Następnie sprawdzono czy dane wyświetlają się w prawidłowy sposób w tabeli za pomocą polecenia data.head():

```
#sprawdzenie czy dane podczytują się w odpowiedni sposób
data.head()
```

Program wyświetla pierwszych 5 rekordów bazy z 32 kolumnami (id pacjentki, diagnoza, 30 cech dla guza).

	Id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean
0	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864
1	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990
2	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390
3	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280
4	843786	M	12.45	15.70	82.57	477.1	0.12780	0.17000

5 rows × 32 columns

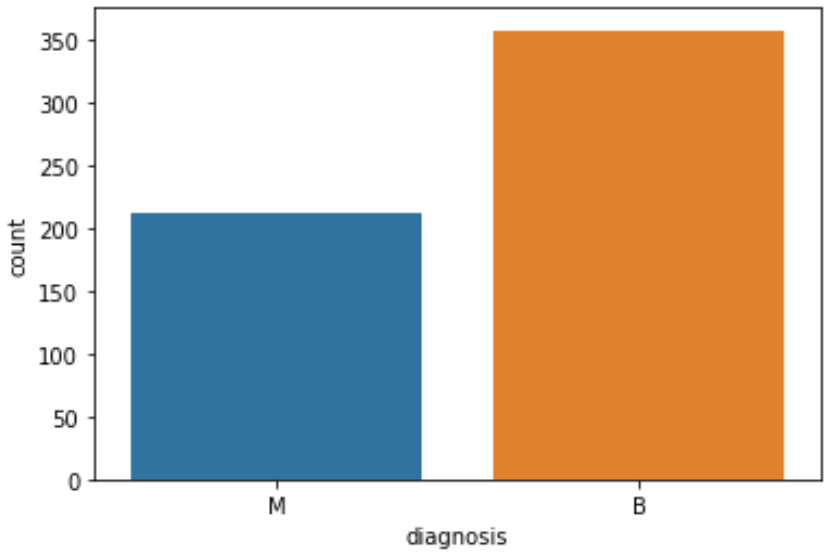
[…]

4.3 Podstawowa analiza za pomocą statystyki opisowej

Na początku podliczono ile w zbiorze znajduje się przypadków złośliwych i łagodnych raka piersi w następujący sposób:

```
#podliczenie ile przypadkow guzów złośliwych, a ile łagodnych
diagnosis_all = list(data.shape)[0]
diagnosis_categories = list(data['diagnosis'].value_counts())
print("\n \t Zbiór danych posiada {} diagnozowanych guzów, {}
łagodnych i {} złośliwych.".format(diagnosis_all,
diagnosis_categories[0], diagnosis_categories[1]))

#wykres słupkowy
y=data.diagnosis
ax = sns.countplot(y,label="Count")
B, M = y.value_counts()
```

```
    Zbiór danych posiada 568 diagnozowanych guzów, 357 łagodnych
i 211 złośliwych.
```



Następnie za pomocą metody data.describe() wyświetlono 3 tabele dla wartości mean (średniej arytmetycznej), se (odchyleń standardowych) i worst (najgorszych/ największych), w których pokazano liczbę rekordów, średnią arytmetyczną, odchylenie standardowe, wartości minimalne, 25%, 50%, 75% i maksymalne wg danej cechy.

```
#zmiana wielkosci komorki w Jupyterze
from IPython.core.display import display, HTML
display(HTML("<style>.container { width:100% !important; }</style>"))

#wyświetlenie podstawowych statystyk dla danych liczbowych MEAN
data_mean = list(data.columns[1:12])
data[data_mean].describe()
```

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave_points_mean	symmetry_mean	fractal_dimension_mean
count	568.000000	568.000000	568.000000	568.000000	568.000000	568.000000	568.000000	568.000000	568.000000	568.000000
mean	14.120491	19.305335	91.914754	654.279754	0.096321	0.104036	0.088427	0.048746	0.181055	0.062770
std	3.523416	4.288506	24.285848	351.923751	0.014046	0.052355	0.079294	0.038617	0.027319	0.007035
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	0.049960
25%	11.697500	16.177500	75.135000	420.175000	0.086290	0.064815	0.029540	0.020310	0.161900	0.057697
50%	13.355000	18.855000	86.210000	548.750000	0.095865	0.092525	0.061400	0.033455	0.179200	0.061515
75%	15.780000	21.802500	103.875000	782.625000	0.105300	0.130400	0.129650	0.073730	0.195625	0.066120
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	0.097440

```
#wyświetlenie podstawowych statystyk dla danych liczbowych SE
data_se = list(data.columns[12:22])
data[data_se].describe()
```

	radius_se	texture_se	perimeter_se	area_se	smoothness_se	compactness_se	concavity_se	concave_points_se	symmetry_se	fractal_dimension_se
count	568.000000	568.000000	568.000000	568.000000	568.000000	568.000000	568.000000	568.000000	568.000000	568.000000
mean	0.403958	1.217402	2.855984	40.138025	0.007042	0.025437	0.031855	0.011789	0.020526	0.003791
std	0.276038	0.551979	2.009288	45.282406	0.003005	0.017897	0.030199	0.006173	0.008264	0.002646
min	0.111500	0.360200	0.757000	6.802000	0.001713	0.002252	0.000000	0.000000	0.007882	0.000895
25%	0.232375	0.833150	1.605000	17.850000	0.005166	0.013048	0.015063	0.007634	0.015128	0.002244
50%	0.323950	1.109500	2.285500	24.485000	0.006374	0.020435	0.025875	0.010920	0.018725	0.003162
75%	0.477325	1.474250	3.336750	45.017500	0.008151	0.032217	0.041765	0.014710	0.023397	0.004526
max	2.873000	4.885000	21.980000	542.200000	0.031130	0.135400	0.396000	0.052790	0.078950	0.029840

```
#wyświetlenie podstawowych statystyk dla danych liczbowych WORST
data_worst = list(data.columns[22:32])
data[data_worst].describe()
```

	radius_worst	texture_worst	perimeter_worst	area_worst	smoothness_worst	compactness_worst	concavity_worst	concave_points_worst	symmetry_worst	fractal_dimension_worst
count	568.00000	568.000000	568.000000	568.000000	568.000000	568.000000	568.000000	568.000000	568.000000	568.000000
mean	16.25315	25.691919	107.125053	878.578873	0.132316	0.253541	0.271414	0.114341	0.289776	0.083884
std	4.82232	6.141662	33.474687	567.846267	0.022818	0.156523	0.207989	0.065484	0.061508	0.018017
min	7.93000	12.020000	50.410000	185.200000	0.071170	0.027290	0.000000	0.000000	0.156500	0.055040
25%	13.01000	21.095000	84.102500	514.975000	0.116600	0.146900	0.114475	0.064730	0.250350	0.071412
50%	14.96500	25.425000	97.655000	685.550000	0.131300	0.211850	0.226550	0.099840	0.282050	0.080015
75%	18.76750	29.757500	125.175000	1073.500000	0.146000	0.337600	0.381400	0.161325	0.317675	0.092065
max	36.04000	49.540000	251.200000	4254.000000	0.222600	1.058000	1.252000	0.291000	0.663800	0.207500

Podsumowując, na tej podstawie stwierdzono obecność 357 przypadków raka łagodnego i 211 złośliwego wśród pacjentek. Cechy opisujące nowotwory są bardzo różne o szerokich zakresach. Zaobserwowane wzrost wszystkich wartość wraz ze zwiększającymi się cechami opisującymi wielkość raka.

5 Wizualizacja danych

W tym rozdziale wykonano wizualizacje zbioru *Breast Cancer Wisconsin (Diagnostic) Data Set*. Przedstawiono wyniki jako macierz korelacji/wykres rozrzutu, wykres ramka-wąsy i wykres słupkowy.

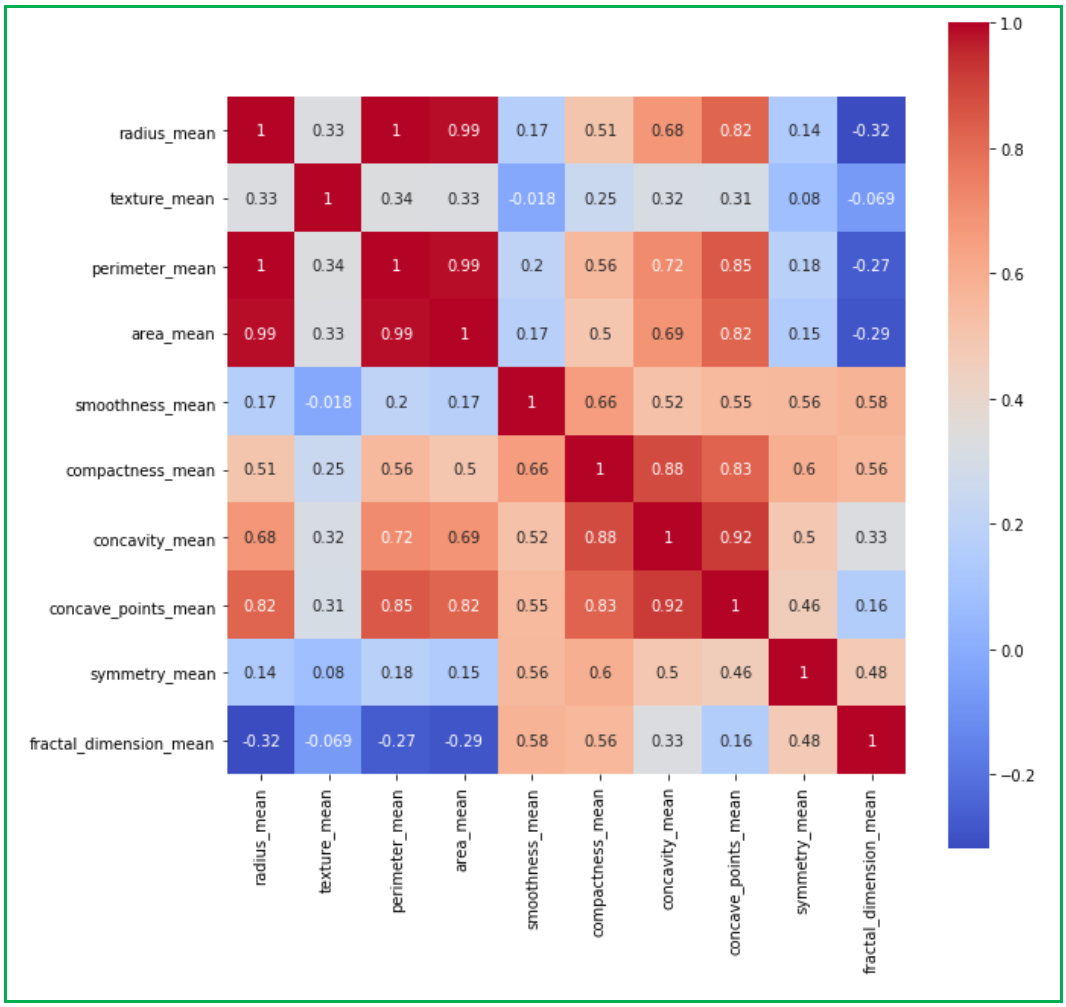
5.1 Macierz korelacji

Macierz korelacji to macierze, której elementy są wartościami współczynników korelacji dla odpowiednich par zmiennych losowych. Powinna ona spełniać 5 kryteriów, po pierwsze to być macierzą kwadratową i jej wartości muszą się mieścić w przedziale <-1, 1>. Dodatkowo wszystkie elementy wszystkie elementy leżące na głównej przekątnej tej macierzy równe są 1. Macierz korelacji jest macierzą symteryczną i jej wyznacznik leży w zakresie <0,1> [13].

Dla zbioru *Breast Cancer Wisconsin (Diagnostic) Data Set* składającego się z wartości średnich, odchylenia standardowego oraz najgorszych wyników danej cechy obliczono macierze korelacji oraz wykreślono wykresy rozrzutu. Dane zostały pokazane w trzech grupach: MEAN (wartości średnie dla 10 cech), SE (odchylenie standardowe) i WORST (najgorsze wyniki guza).

Poniżej przedstawiono kod implementacji powstania macierzy dla wartości MEAN oraz wynik działania kodu:

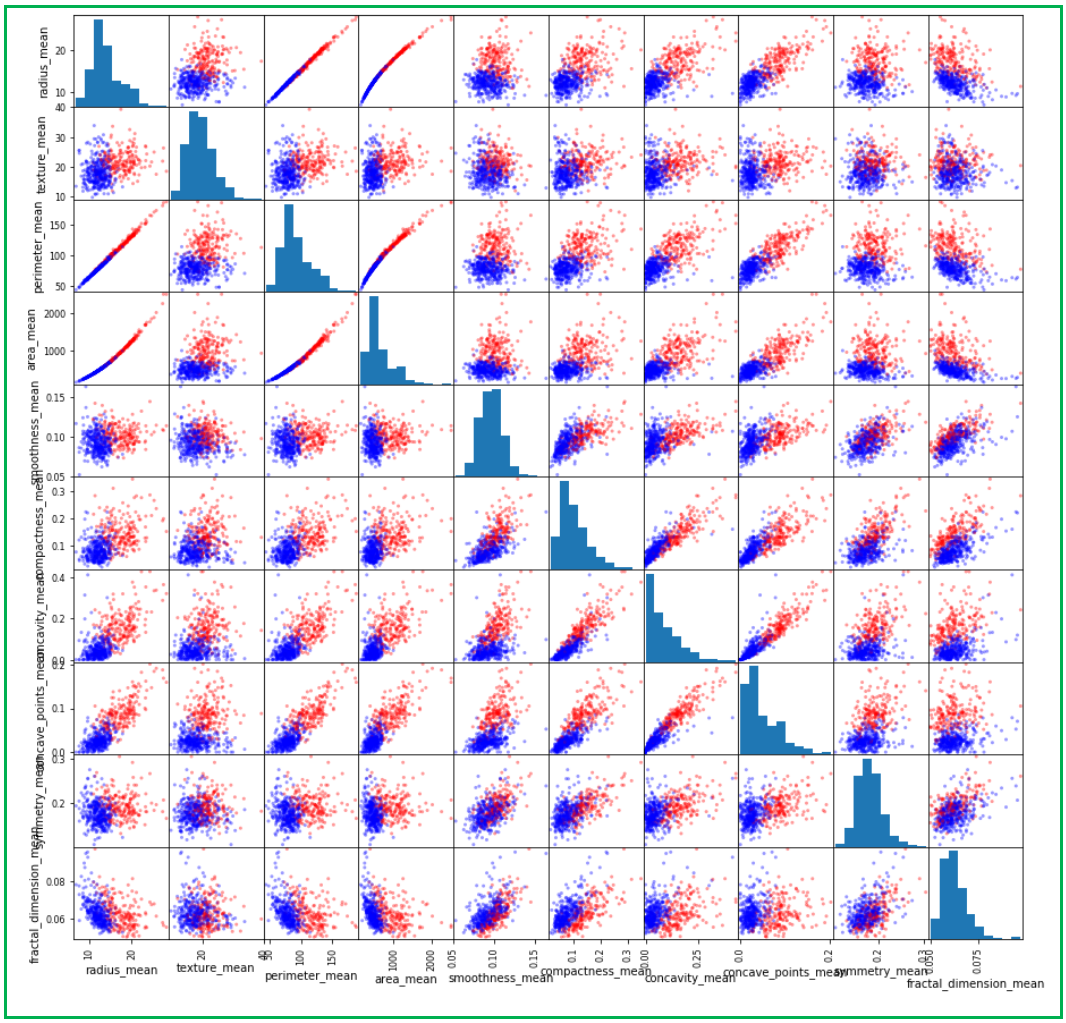
```
# Macierz korelacji  dla danych MEAN
features_mean = list(data.columns[1:12])
plt.figure(figsize=(10,10))
sns.heatmap(data[features_mean].corr(), annot=True, square=True,
cmap='coolwarm')
plt.show()
```



Największą korelację możemy zaobserwować miedzy obszarem, obwodem i promieniem guza. Jest to logiczne, bo wraz z namnażaniem komórek rakowych i wzrostem ich ilości zwiększa się obszar jaki zajmują. Również widać silną dodatnią korelację między area, perimeter, radius a compactness, concavity. Wraz ze wzrostem promienia, obszaru i obwodu guza rośle liczba jego punktów wklęsłości na powierzchni.

Poniżej przedstawiono kod implementacji wykresu rozrzutu dla wartości MEAN oraz output:

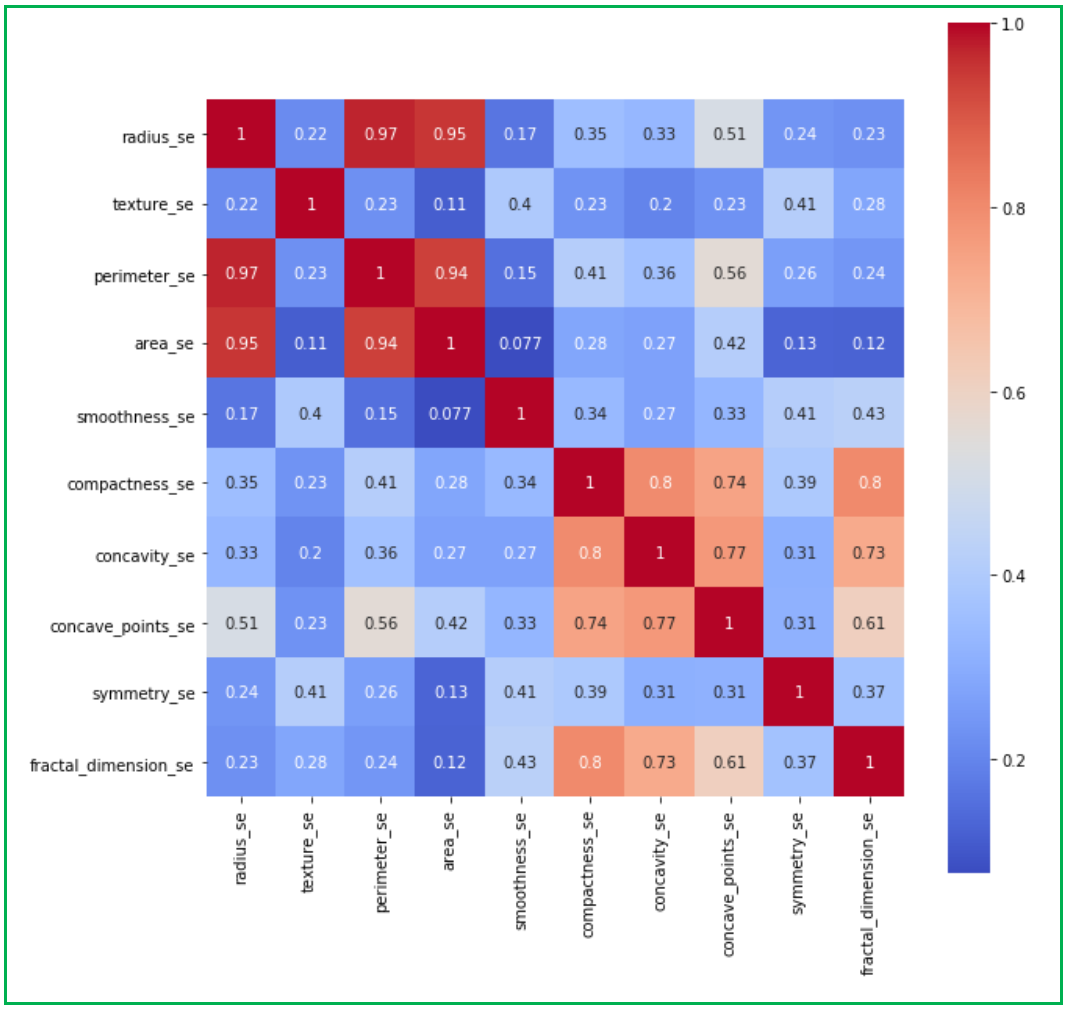
```
# Wykres rozrzutu dla danych MEAN
color_dic = {'M':'red', 'B':'blue'}
colors = data['diagnosis'].map(lambda x: color_dic.get(x))
sm = pd.plotting.scatter_matrix(data[features_mean], c=colors,
alpha=0.4, figsize=(15,15))
plt.show()
```



Na wykresie rozrzutu czerwonym kolorem zostały zaznaczone wyniki dla raka złośliwego (M), a niebieskim – dla raka łagodnego (B). W tym przypadku również można zaobserwować te same dodatnie korelacje i jeden wniosek, że wraz ze wzrostem guza wrasta jego wklęsłość.

Następnie wykonano macierz korelacji i wykres rozrzutu dla wartości odchylenia standardowego SE w poniższy sposób:

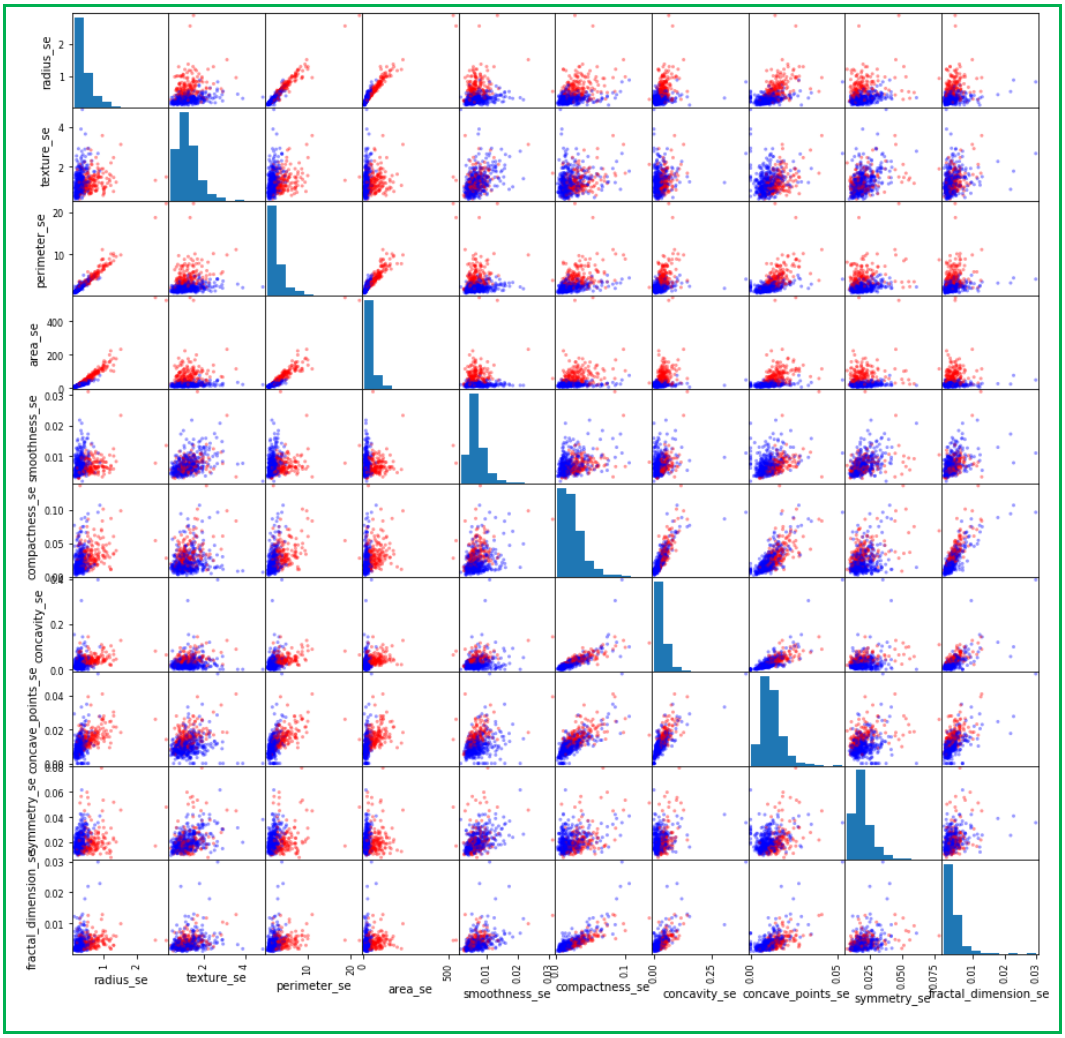
```
# Macierz korelacji  dla danych SE
features_se = list(data.columns[12:22])
plt.figure(figsize=(10,10))
sns.heatmap(data[features_se].corr(), annot=True, square=True,
cmap='coolwarm')
plt.show()
```



W tym przypadku macierzy dla SE korelacje są na podobnym poziomie co dla wartości MEAN dla zależności między promieniem, obwodem a obszarem. Natomiast można zaobserwować spadek współczynników korelacji dla zależności między wklęsłością a punktami wklęsłymi.

Poniżej przedstawiono kod dla wykresu rozrzutu dla danych odchylenia standardowego:

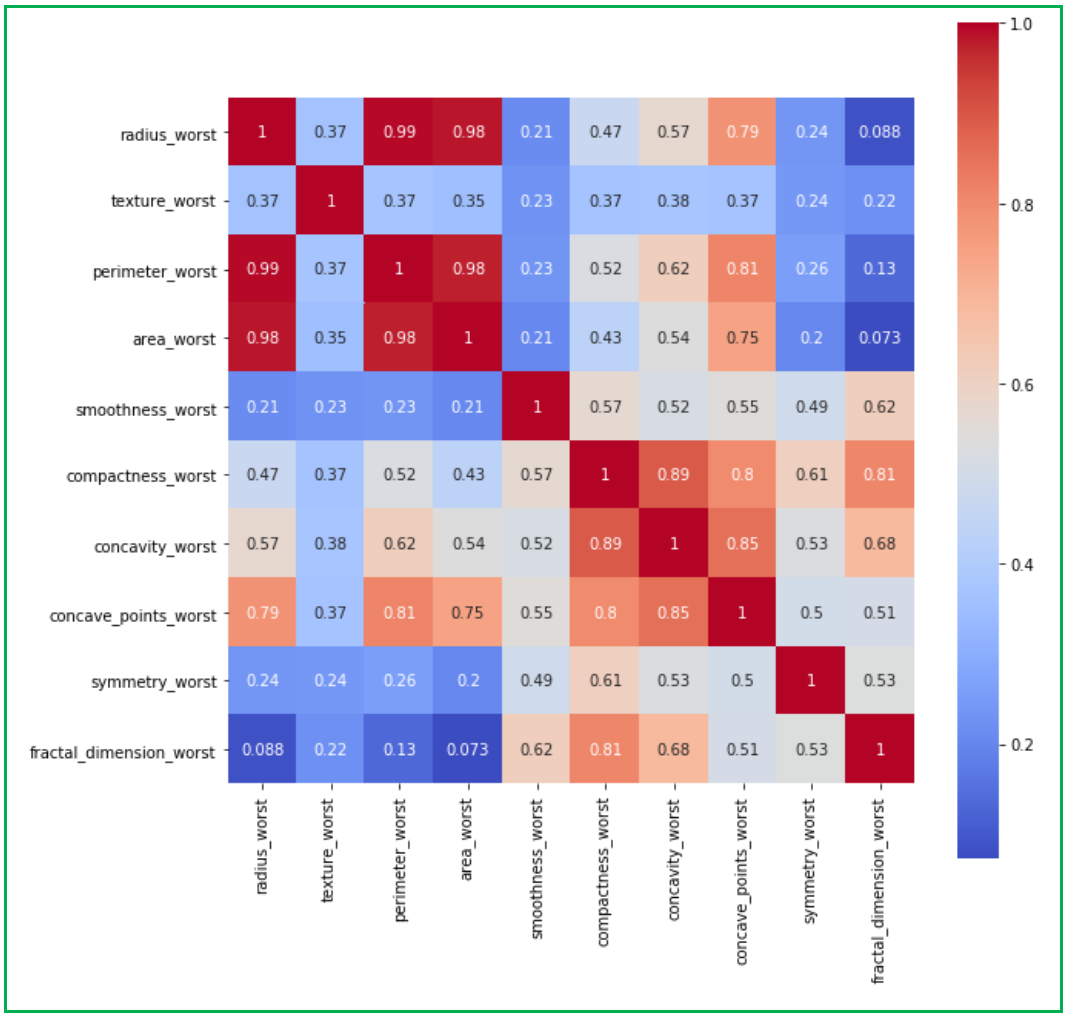
```
# Wykres rozrzutu dla danych SE
color_dic = {'M':'red', 'B':'blue'}
colors = data['diagnosis'].map(lambda x: color_dic.get(x))
sm = pd.plotting.scatter_matrix(data[features_se], c=colors,
alpha=0.4, figsize=(15,15))
plt.show()
```



Wykres rozrzutu potwierdził zmniejszenie się wartości współczynników korelacji między postawionymi hipotezami w porównaniu z wartościami MEAN.

Również dla danych WORST wykonano macierz korelacji i wykres rozrzutu w następujący sposób:

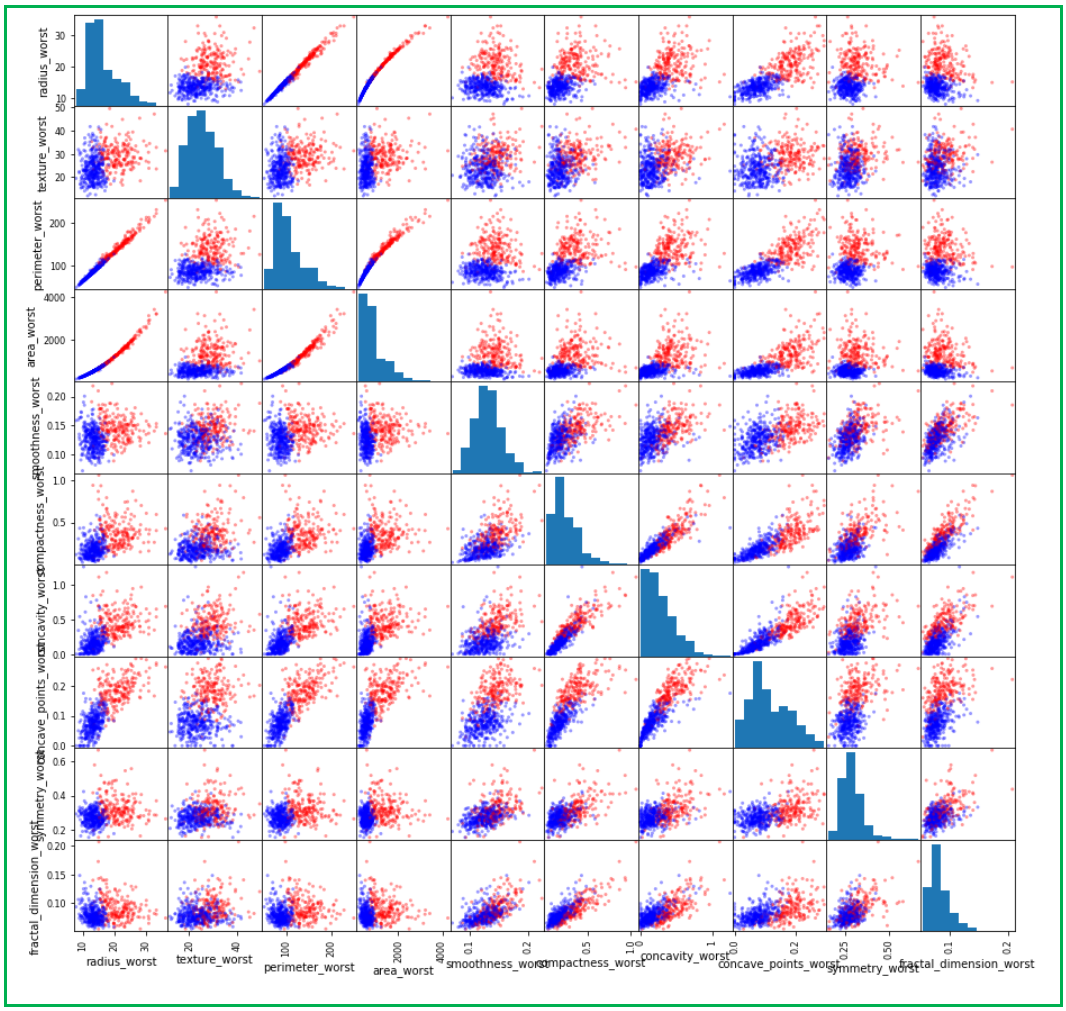
```
# Macierz korelacji  dla danych WORST
features_worst = list(data.columns[22:32])
plt.figure(figsize=(10,10))
sns.heatmap(data[features_worst].corr(), annot=True,
square=True, cmap='coolwarm')
plt.show()
```



Otrzymana macierz korelacji dla najgorszych wartości zbioru posiada bardzo podobne zależności do wartości średnich. Można zaobserwować, że najgorszy/ największy promień są skorelowane z obszarem i obwodem. Dodatkowo widać zależności między parametrami wklęsłości między sobą.

NA podstawie wyników macierzy korelacji stworzono wykres rozrzutu dla najgorszych/ największych wartości:

```
# Wykres rozrzutu dla danych WORST
color_dic = {'M':'red', 'B':'blue'}
colors = data['diagnosis'].map(lambda x: color_dic.get(x))
sm = pd.plotting.scatter_matrix(data[features_worst], c=colors,
alpha=0.4, figsize=(15,15))
plt.show()
```



Tak jak już wcześniej wspomniano, wyniki korelacji uzyskane dla wartości najgorszych są bardzo zbliżone do MEAN.

Podsumowując, na podstawie macierzy korelacji i wykresów rozrzutu udało się znaleźć zależność miedzy promienie, obszarem i obwodem guza. Można śmiało powiedzieć, że wraz ze wzrostem promienia nowotworu, wzrasta obwód i obszar raka piersi. Zauważono również, że wraz ze wzrostem liczby wklęsłych to wzrasta wklęsłość i zwartość guza.

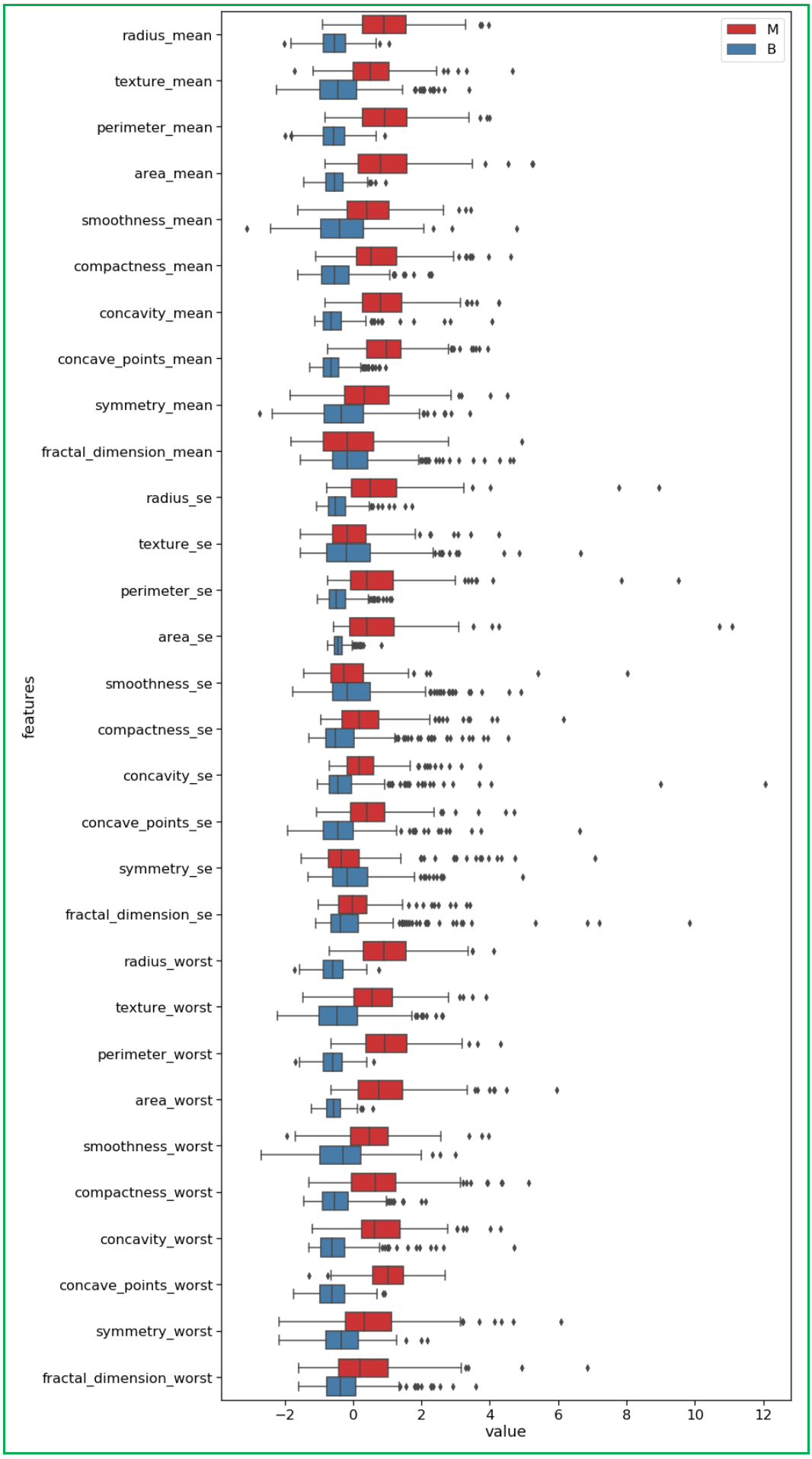
5.2 Wykres ramka-wąsy

Wykres ramka-wąsy, inaczej zwany wykresem skrzynkowym lub pudełkowym (z ang. *box-plot*). Opracowane są w oparciu o wartości statystyk opisowych, dlatego ich zastosowanie służy do przedstawienia cech liczbowych. Stanowią prostą formę prezentacji graficznej rozkładu cechy statystycznej. Pozwalają na ujęcie na jednym rysunku wszystkich wiadomości, które dotyczą położenia, kształtu a także rozkładu empirycznego badanej cechy. Charakteryzuje je duża przejrzystość i zwięzłość. Wąsy reprezentują odległość wartości maksymalnej i minimalnej od pudełka. Długość pudełka natomiast to rozstał ćwiartkowy – obejmuje 50% środkowych obserwacji. Kreska dzieląca pudełko na pół oznacza wartość mediany [14].

Wiedząc już czym jest wykres box-plot to przystąpiono do jego implementacji i stworzono dwa wykresy dla każdej cechy, gdzie czerwone pudełko oznacza raka złośliwego, a niebieskie – łagodnego.

Poniżej przedstawiono ciąg dalszy kodu które obejmuje stworzenie wykresów ramka-wąsy.

```
#Wykresy ramka-wąsy
list = ['Id','diagnosis']
features = data.drop(list,axis = 1,inplace = False)

stdX = (features - features.mean()) / (features.std())
data_st = pd.concat([y,stdX.iloc[:,:]],axis=1)
data_st = pd.melt(data_st,id_vars="diagnosis",
                    var_name="features",
                    value_name="value")

plt.figure(figsize=(12,30))
sns.set_context('notebook', font_scale=1.5)
sns.boxplot(x="value", y="features", hue="diagnosis",
data=data_st, palette='Set1')
plt.legend(loc='best')
```


M
B
radius_mean
texture_mean
perimeter_mean
area_mean
smoothness_mean
compactness_mean
concavity_mean
concave_points_mean
symmetry_mean
fractal_dimension_mean
radius_se
texture_se
perimeter_se
area_se
smoothness_se
compactness_se
concavity_se
concave_points_se
symmetry_se
fractal_dimension_se
radius_worst
texture_worst
perimeter_worst
area_worst
smoothness_worst
compactness_worst
concavity_worst
concave_points_worst
symmetry_worst
fractal_dimension_worst
features
−2
0
2
4
6
8
10
12
value

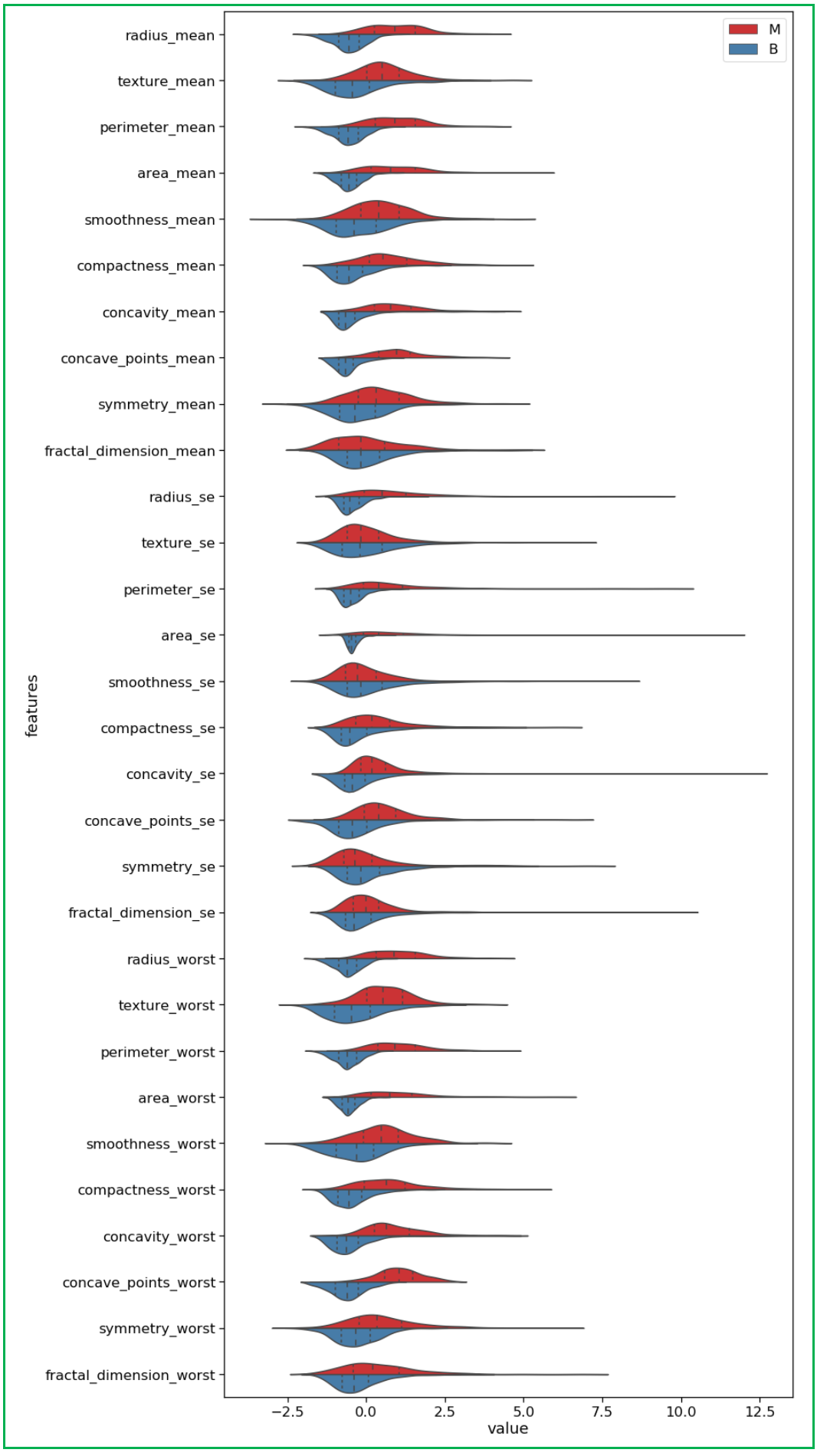
Na podstawie sporządzonych wykresów ramka-wąsy można zauważyć, że wartości cech dla raka złośliwego mają większe wartości od raka łagodnego. Złośliwy nowotwór piersi posiada większy promień, obwód i zajmowany obszar oraz bardziej nierównomierną powierzchnie o małej symetrii od raka łagodnego. Można podejrzewać zatem, że komórki raka M mogą namnażać się szybciej od komórek B oraz powodować gorsze szkody w ludzkim organizmie oraz większą ilość przypadków śmiertelnych.

5.3 Wykres skrzypcowy

Wykres skrzypcowy (ang. *violin plot*) pozwala wizualizować jednocześnie gęstość prawdopodobieństwa oraz rozkład dla kilku grup jednocześnie. Każde „skrzypce" przedstawiają grupę lub zmienną. Kształt skrzypiec reprezentuje tak naprawdę oszacowanie gęstości zmiennej: im więcej punktów danych w określonym zakresie, tym większe skrzypce będą w tym miejscu. Skrzypce są szczególnie przystosowane, gdy ilość danych jest ogromna, a pokazanie indywidualnych obserwacji staje się niemożliwe. Biała kropka na środku to wartość mediany, a gruby czarny pasek na środku reprezentuje zakres międzykwartylowy. Cienka czarna linia przedłużona od niej reprezentuje górne i dolne wartości w danych [15].

Tak jak w przypadku wykresu *box-plot*, tutaj również przedstawiono dla każdej cechy wykres, gdzie czerwony oznacza raka złośliwego i niebieski – raka łagodnego. Kod przedstawiono poniżej.

```
# Wykresy skrzypcowe

plt.figure(figsize=(12,30))
sns.set_context('notebook', font_scale=1.5)
sns.violinplot(x="value", y="features", hue="diagnosis",
data=data_st,split=True,
               inner="quart", palette='Set1')
plt.legend(loc='best')
```


M
B
radius_mean
texture_mean
perimeter_mean
area_mean
smoothness_mean
compactness_mean
concavity_mean
concave_points_mean
symmetry_mean
fractal_dimension_mean
radius_se
texture_se
perimeter_se
area_se
smoothness_se
compactness_se
concavity_se
concave_points_se
symmetry_se
fractal_dimension_se
radius_worst
texture_worst
perimeter_worst
area_worst
smoothness_worst
compactness_worst
concavity_worst
concave_points_worst
symmetry_worst
fractal_dimension_worst
features
−2.5
0.0
2.5
5.0
7.5
10.0
12.5
value

Spoglądając na wykresy skrzypcowe dla wszystkich 30 cech można zauważyć kilka ciekawych wniosków. Po pierwsze w przypadku wykresów dla wartości MEAN widać, że gęstość prawdopodobieństwa dla przypadków B zlokalizowana jest przy mniejszych wartościach od raka M. Praktycznie to samo pokazały wykresy ramka-wąsy. Po drugie, wartości dla raka złośliwego posiadają szerszy zakres skrzypiec od raka łagodnego, co świadczy również o mniejszych wartościach dla raka łagodnego. Dodatkowo dla wykresów cech WORST widać, że przypadki najgorsze dla raka złośliwego mają większe wartości od najgorszych przypadków raka łagodnego. Potwierdza to za zatem postawioną wcześniej hipotezę, że rak złośliwy ma większe rozmiary od łagodnego i powoduje większą śmiertelność.

6 Analiza głównych składowych

Analiza głównych składowych (PCA) to najbardziej popularny algorytm redukcji wymiarów, która polega na rzutowaniu danych do przestrzeni o mniejszej liczbie wymiarów tak, aby jak najlepiej zachować strukturę danych. Dokładna analiza składowych głównych umożliwia wskazanie tych zmiennych początkowych, które mają duży wpływ na wygląd poszczególnych składowych głównych czyli tych, które tworzą grupę jednorodną.

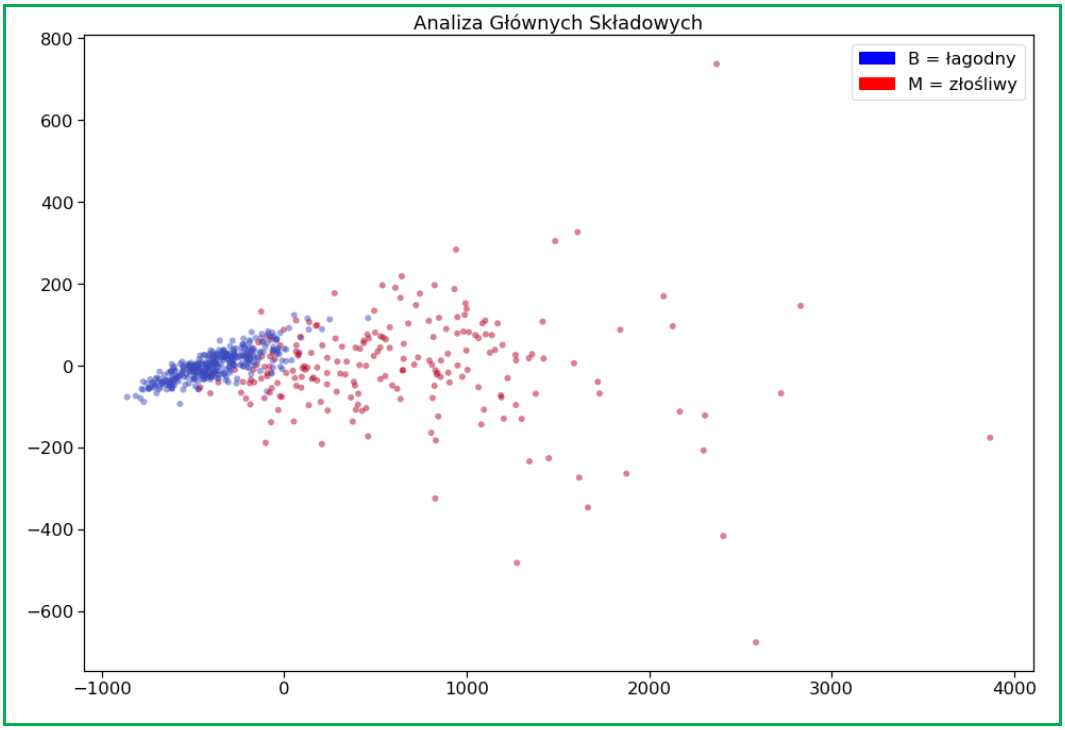
Analiza PCA opiera się o wyznaczenie osi zachowującej największą wartość wariancji i zbioru uczącego. Idee tworzenia kolejnych składowych polegają na tym, że kolejne składowe nie są skorelowane ze sobą oraz mają na celu zmaksymalizować zmienność, która nie została wyjaśniona przez poprzednią składową [16].

Jak działa algorytm PCA?

1. W pierwszym kroku liczy średnie dla obu wierszy w zbiorze danych.
2. Krok ten polega na odjęciu od macierzy wejściowej średnich obliczonych w punkcie pierwszym.
3. Następnie wyznacza macierz kowariancji.
4. W kroku czwartym oblicza wartości własne macierzy kowariancji.
5. Na tym etapie dochodzi do zwężenia wymiaru przestrzeni. Wybierane są wartości własne, które są największe, co ma na celu minimalizację strat informacji podczas rzutowania danych na mniejsza liczę wymiarów.
6. Następnie są wyznaczane wektory własne, np. poprzez zastosowanie algorytmu eliminacji Gaussa.
7. Ostatecznie dochodzi do rzutowania na wektory własne, czyli wyznaczeniu nowego punktu w przestrzeni (mnożenie macierzy) [17].

Poniżej przedstawiono kod dla analizy PCA, w której podzielono dane na dwa komponenty złośliwy M i łagodny B, czyli liczbę n czynników w nowej przestrzeni (funkcja n_components):

```
#Pomięniecie kolumnn 'Id' i 'diagnosis'
df=data.drop('Id',axis=1)
x = df.drop('diagnosis', axis = 1 )
x.head()

y_df= pd.get_dummies(y,drop_first=True) # porzucenie kolumny o
nazwie diagnosis' oraz kolumn 0 i 1
y_df.head()
y_df=y_df['M']

# Konwertowanie danych do tablicy
X = x.values

# Wywołanie metody PCA o nazwie n_components = 2
pca = PCA(n_components=2)
pca_2d = pca.fit_transform(X)

# Wykres PCA
plt.figure(figsize = (16,11))
plt.scatter(pca_2d[:,0],pca_2d[:,1], c = y_df,
            cmap = "coolwarm", edgecolor = "None", alpha=0.5,);
plt.title('Analiza Głównych Składowych');

rects=[]
rects.append(mpatches.Patch(color='blue', label='B = łagodny'));
rects.append(mpatches.Patch(color='red', label='M = złośliwy'));
plt.legend(handles=rects);
```



Za pomocą analizy PCA udało się podzielić cechy na dwa wymiary. Niebiskim kolorem zaznaczono raka łagodnego, a czerwonym – złośliwego. Na wykresie jednej głównej składowej od drugiej widać, że rak B stworzył jedno skupisko, natomiast wyniki PCA dla raka złośliwego są rozrzucone w całym zakresie PC1 i PC2. Oznacza to, że cechy opisujące raka łagodnego powtarzają się i są do siebie podobne. Przez co jest większe prawdopodobieństwo jego przewidzenia, bo cechą są bardziej objaśnione. Rak złośliwy M posiada cechy go opisujące w większych zakresach, które nie tworzą wyraźnego skupiska. Każdy przypadek raka złośliwego znacząco różni się od poprzednich przez co może być trudniejszy w zdiagnozowaniu.

7 Analiza t-SNE

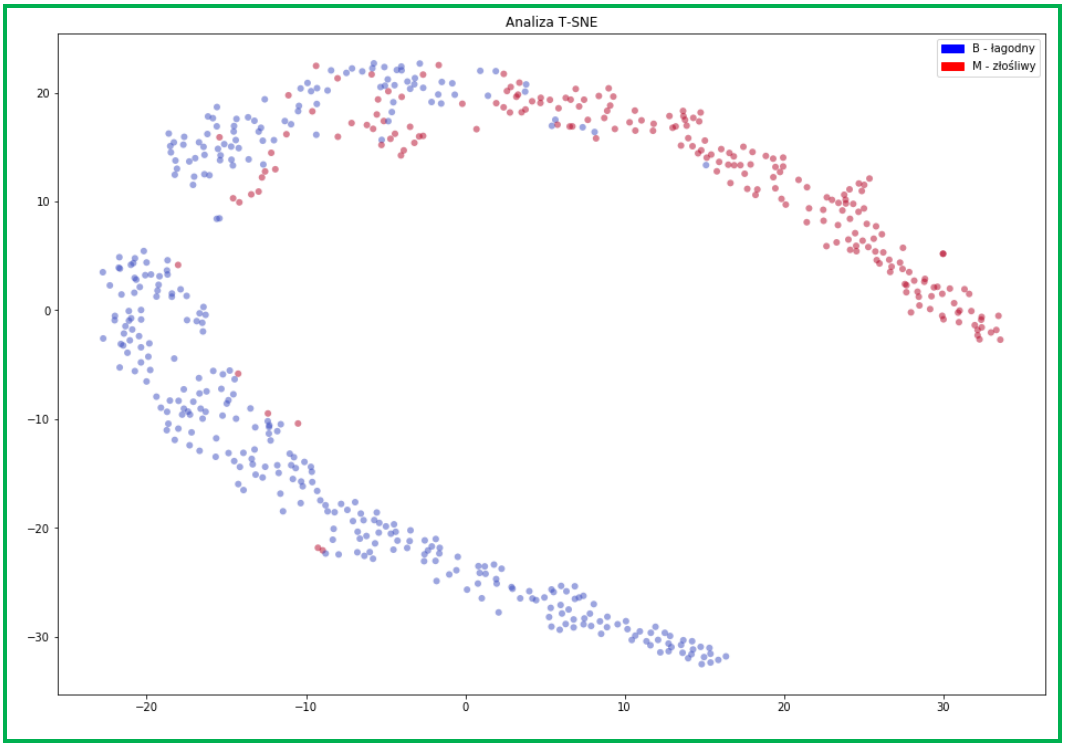
Algorytm t-SNE oblicza miarę prawdopodobieństwa miedzy parami w przestrzeni o dużych wymiarach oraz przestrzeni o małych wymiarach. Kolejno optymalizuje wcześniej wspomniane miary prawdopodobieństwa.

Jak działa algorytm t-SNE?

1. Pierwszym krokiem jest wyliczenie podobieństw wszystkich naszych danych. Określa zasięg w ramach którego patrzy na punkty (kula, najczęściej zakres parametru 5-50). Algorytm wybiera pierwszy punkt i liczy odległość miedzy nim a drugim punktem w tym samym wymiarze. Na wykresie rozkładu normalnego zostaje zaznaczony punk policzonej odległości. Dzięki zastosowaniu rozkładu normalnego widać, że punkty oddalone od siebie mają małe prawdopodobieństwo, a znajdujące się blisko siebie to duże. Taka operacja jest powtarzana dla wszystkich punktów w zasięgu. Mając wyliczone te dystanse naniesione na rozkład normalny t-SNE zamienia je na zestaw prawdopodobieństw wszystkich punktów. Operacja jest powtarzana dla wszystkich punktów w zbiorze.
2. Drugi krok to wyliczenie losowych podobieństw. Losowana jest projekcja w mniejszym wymiarze (np. zmiana 3-wymiarów na 2). Algorytm stosuje test t-studenta o jednym stopniu swobody, który kolejno wylicza wszystkie odległości miedzy punktami i wylicza prawdopodobieństwo oraz drugi zestaw prawdopodobieństwo w przestrzeni niskiego wymiaru.
3. Ostatnim krokiem jest odzwierciedlenie zestawów prawdopodobieństw. Aktualnie zależy nam by prawdopodobieństwa z niższego wymiaru reprezentowały te z wyższego. Różnicę między rozkładami prawdopodobieństwa przestrzeni mierzymy za pomocą dywergencji Kullbacka-Lieblera (KL). Daje ono podejście asymetryczne, które skutecznie porównuje wartości obu macierzy prawdopodobieństw [18].

Poniżej przedstawiono kod dla analizy t-SNE z liczbą komponentów równą dwa, czyli liczbę wymiarów:

```
# Wywołanie metody T-SNE o nazwie n_components = 2
tsne= TSNE(n_components=2)
tsne_2d=tsne.fit_transform(X)

# Wykres T-SNE
plt.figure(figsize = (16,11))
plt.scatter(tsne_2d[:,0],tsne_2d[:,1], c = y_df,
            cmap = "coolwarm", edgecolor = "None", alpha=0.5,);
plt.title('Analiza T-SNE');

rects=[]
rects.append(mpatches.Patch(color='blue', label='B - łagodny'));
rects.append(mpatches.Patch(color='red', label='M - złośliwy'));
plt.legend(handles=rects);
```



T-SNE poradził sobie lepiej w odróżnieniu i podziale raka złośliwego oraz łagodnego na klastery w porównaniu z Analizą Głównych Składowych (PCA). Część przypadków nowotworu B znalazła się w grupie M co może nasuwać myśl, że część raka piersi zdiagnozowanego jako rak łagodny mogła okazać się przypadkiem złośliwym, ponieważ posiadała podobne wartości cech, które mogły być charakterystyczne dla grupy M. Najprawdopodobniej wyniki B w grupie raka złośliwego to ciężkie (WORST) przypadki raka łagodnego.

8 Modele uczenia maszynowego

Dla zbioru *Breast Cancer Wisconsin (Diagnostic) Data Set* zbudowano pięć modeli z wykorzystaniem narzędzi Machine Learning., tj. regresję logistyczną, las losowy, drzewa decyzyjne, K najbliższych sąsiadów i sieci neuronowe w celu klasyfikacji nowotworu łagodnego i złośliwego.

8.1 Regresja logistyczna

Model regresji logistycznej jest przypadkiem uogólnienia modelu liniowego. Znajduje zastosowanie kiedy zmiena jest dychotomiczna, czyli przyjmuje dwie wartości, np. sukces i porażka lub kobieta i mężczyzna. Funkcja logistyczna to krzywa w kształcie litery S, która może przyjąć wartość liby rzeczywistej w zakresie od 0 do 1. Wzór funkcji logistycznej przedstawiono poniżej:

$$y(x) = \frac{1}{1 + e^{-x}}$$

Gdzie e to podstawa logarytmów naturalnych (liczba Eulera), a x jest rzeczywistą wartością liczbową.

Wynik regresji logistycznej, podobnie jak regresji liniowej, można przedstawić za pomocą równania. Wartości wejściowe x są łączone liniowo za pomocą wag lub wartości współczynników, aby przewidzieć wartość wyjściową y. Kluczową różnicą w stosunku do regresji liniowej jest to, że modelowana wartość wyjściowa jest wartością binarną (0 lub 1).

Zakładając, że wartość wejściowa ma postać b^0+b^1*x. Równaniem regresji logistycznej będzie $p(class = 0) = \frac{1}{1+e^{-(b0+b1*x)}}$. Wynikiem algorytmu jest prawdopodobieństwo wyznaczone przy użyciu funkcji logistycznej zdefiniowanej wyżej. Model regresji logistycznej pobiera dane wejściowe o wartościach rzeczywistych i dokonuje przewidywania prawdopodobieństwa wejścia należącego do klasy domyślnej (klasa 0). Jeśli prawdopodobieństwo wynosi więcej niż 0,5, możemy przyjąć wyjście jako prognozę dla klasy domyślnej (klasa 0), w przeciwnym razie przewidywanie jest dla drugiej klasy (klasa 1) [19].

Na samym początku przed stworzeniem modeli podzielono dane na zbiór uczący i zbiór walidacyjny w następujący sposób:

```
# Odzielenie danych w zbiorze uczącym i walidacyjnym
df_train, df_test = train_test_split(df, test_size = 0.3)
x_train=df_train.drop('diagnosis',axis=1)
x_test=df_test.drop('diagnosis',axis=1)
y_train=df_train['diagnosis']
y_test=df_test['diagnosis']
```

Następnie przystąpiono do stworzenia modelu oraz jego trenowania. Poniżej przedstawiono kod implementacji modelu regresji logistycznej dla zbioru danych:

```
# Stworzenie modelu
model_logistic_regression = LogisticRegression()

# Dopasowanie modelu
model_logistic_regression.fit(X=x_train,y=y_train)
```

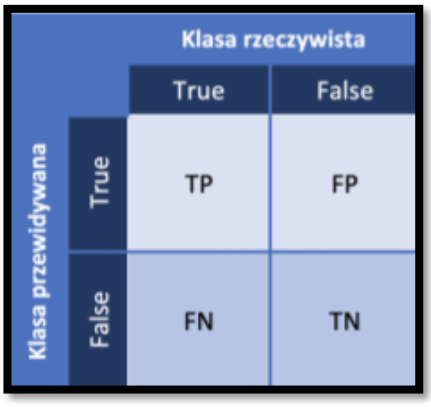
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

Model po wytrenowaniu gotowy jest do klasyfikacji nowych tekstów za pomocą funkcji .predict() przyjmującej jako parametr zbiór testowy (x_test), który powinien mieć ten sam format, co zbiór użyty podczas wywoływania funkcji .fit():

```
# Predykcja
prediction_logistic_regression =
model_logistic_regression.predict(x_test)
```

Ewaluację modelu w celu oceny jego jakości przeprowadzono przez wyznaczenie macierzy pomyłek, precyzji, pokrycia, F1 wyniku i dokładności.

Macierz pomyłek to metoda reprezentacji wyników predykcji w problemach klasyfikacyjnych. Jest to tabela z czterema różnymi kombinacjami wartości przewidywanych i rzeczywistych. Poniższy rysunek przedstawia ogólną postać macierzy pomyłek [20].



Rys. 10. Macierz pomyłek.

Oznaczenia na powyższym rysunku macierzy pomyłek oznaczają:

- Prawdziwie pozytywy (TP): gdy rzeczywista klasa danych to P (prawda), a przewidywana jest również P (prawda).

- Prawdziwie negatywny (TN): gdy rzeczywista klasa danych wynosiła N (fałsz), a przewidywana jest również N (fałsz).
- Fałszywie pozytywny (FP): gdy rzeczywista klasa danych wynosiła N (fałsz), a przewidywana to P (prawda).
- Fałszywie negatywy (F): Gdy rzeczywista klasa danych wynosiła P (prawda), a przewidywana wartość to N (fałsz).

Kolejną metodą ewaluacyjną jest precyzja/precision to pozwala przewidzieć czy predykcja jest poprawna. Przykładowo załóżmy, że w zbiorze testowym mamy 1000 wierszy, z czego 300 należy do klasy P. Nasz model przewiduje, że do klasy P należy 450, z czego wszystkie 300 wskazał poprawnie, a 150 oszacował źle. W tym wypadku precyzja wynosi 0,667 [21]. Można określić ją wzorem:

$$Precision = \frac{TP}{TP + FP}$$

Natomiast pokrycie/recall sprawdza prawdopodobieństwo, że model przewidzi daną wartość trafiającą do klasy T (true) [21].

$$Recall = \frac{TP}{TP + FN}$$

Accurancy oznacza dokładność/celność i jest to stosunek poprawnie przewidzianych wartości do łącznej liczby wierszy w zbiorze testowym [21].

$$Accurancy = \frac{TP + TN}{TP + FP + TN + FN}$$

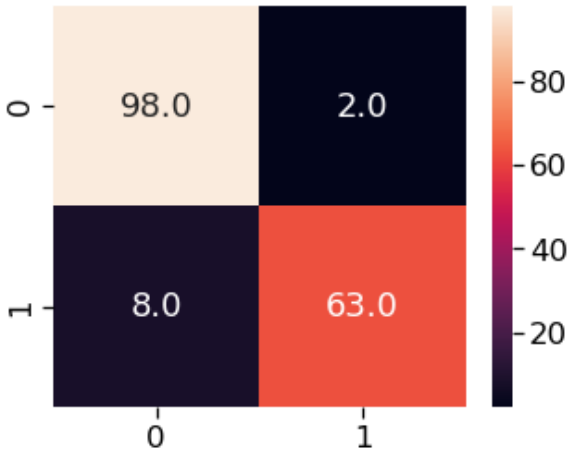
Wynik F1/ F1-score to średnia harmoniczna precyzji i pokrycia zgodna z poniższym wzorem [22]:

$$F1 - score = 2 * \frac{precision * recall}{precision + recall} = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

Poniżej przedstawiono zaimplementowany kod dla ewaluacji modelu, który zawiera raport klasyfikacyjny oraz macierz pomyłek:

```
# Ewaluacja modelu

#Raport klasyfikacyjny
results_logistic_regression =
metrics.classification_report(y_true=y_test,
y_pred=prediction_logistic_regression)
print(results_logistic_regression)

# Macierz pomyłek
confusion_matrix_logistic_regression=metrics.confusion_matrix(y_
true=y_test, y_pred=prediction_logistic_regression)
plt.subplots(figsize=(5, 4))
sns.heatmap(confusion_matrix_logistic_regression, annot=True,
fmt= '.1f')
```

	precision	recall	f1-score	support
B	0.92	0.98	0.95	100
M	0.97	0.89	0.93	71
accuracy			0.94	171
macro avg	0.95	0.93	0.94	171
weighted avg	0.94	0.94	0.94	171



Za pomocą raport klasyfikacji oraz macierzy pomyłek można zauważyć, że model regresji logistycznej uzyskał wysokie i zadawalające wyniki. Model ten wykazał dokładność na poziomie 0.94 w zbiorze testowym, czyli na 100 przypadków raka udało mu się zdiagnozować 94 kobiety poprawnie. Dodatkowo precyzja (poprawność predykcji) dla M wyniosła 0.97 a dla B – 0.92. Natomiast pokrycie, czyli prawdopodobieństwo przewidzenia przez model poprawnego wyniku dla raka złośliwego M wyniosło 0.89, a dla łagodnego B – 0.98. Jeśli chodzi o wynik F1 uzyskano go na poziomie 0.95 dla B i 0.93 dla M.

Na podstawie macierzy pomyłek widać, że 98 przypadków zostało zdiagnozowanych prawdziwie pozytywnie, a 63 uzyskało status fałszywie negatywny.

8.2 Las losowy

Las losowy jest pojęciem, które zostało zaproponowane przez Leo Breimana. Stanowi jeden z najciekawszych oraz najbardziej użytecznych algorytmów uczenia maszynowego. Jest niezwykle popularny w badaniach genetycznych, przewidywaniu aktywności biologicznej cząsteczek czy analizie dokumentów tekstowych – po prostu tam, gdzie mamy do czynienia z dużą liczbą cech.

Las losowy zbudowany jest z wielu klasyfikatorów (drzew decyzyjnych). Dodatkowo każde drzewo budowane jest na innym losowo wybranym podzbiorze

danych uczących. Każdy podział w drzewie jest wybierany jako najlepszy możliwy podział dla losowo wybranego małego podzbioru zmiennych (stąd nazwa Losowy).

Pierwszym krokiem stworzenia modelu lasu losowego jest losowanie ze zwracaniem podzbioru danych (przypadków) z dostępnej próby uczącej. Następnie, powinniśmy stworzyć drzewo dla wylosowanego wcześniej podzbioru. W tym punkcie trzeba sprawdzić, czy dzielony zbiór jest jednorodny oraz czy nie jest zbyt mały, by go podzielić. Należy wylosować pewną liczbę zmiennych objaśniających oraz znaleźć najlepszy podział z wykorzystaniem wylosowanego podzbioru zmiennych. Zbiór ten później dzielimy na dwie części. Jeżeli liczba drzew osiągnie zadane maksimum lub błąd w próbie testowej przestanie maleć, należy zakończyć uczenie. W przeciwnym przypadku, należy wrócić do pierwszego kroku. W ogólności: drzewa "głosują" nad rozwiązaniem, wybór następuje zwykłą większością głosów [23].

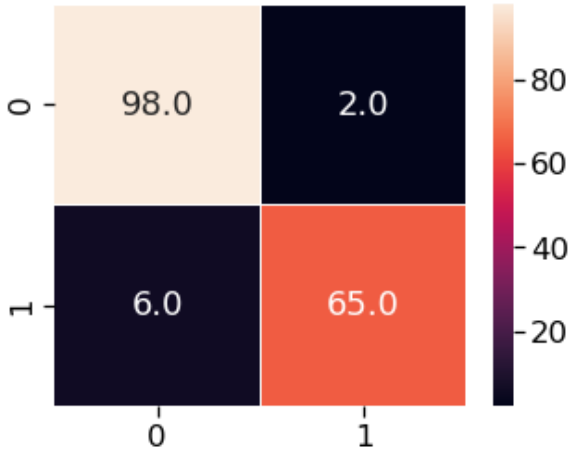
Poniżej przedstawiono model lasu losowego dla zbioru *Breast Cancer Wisconsin (Diagnostic) Data Set:*

```
# Stworzenie modelu
model_random_forest = RandomForestClassifier()

# Dopasowanie modelu
model_random_forest.fit(X=x_train, y=y_train)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

```
# Predykcja
prediction_random_forest = model_random_forest.predict(x_test)

# Ewaluacja modelu

#Raport klasyfikacyjny
results_random_forest =
metrics.classification_report(y_true=y_test,
y_pred=prediction_random_forest)
print(results_random_forest)

# Macierz pomyłek
confusion_matrix_random_forest=metrics.confusion_matrix(y_true=y
_test, y_pred=prediction_random_forest)
f,ax = plt.subplots(figsize=(5, 4))
sns.heatmap(confusion_matrix_random_forest, annot=True,
linewidths=.5, fmt= '.1f',ax=ax)
```

	precision	recall	f1-score	support
B	0.94	0.98	0.96	100
M	0.97	0.92	0.94	71
accuracy			0.95	171
macro avg	0.96	0.95	0.95	171
weighted avg	0.95	0.95	0.95	171



Las losowy wykazał się dokładnością na poziomie 0.95, czyli na 100 kobiet, aż 95 zdiagnozował poprawnie w zbiorze testowym. Precyzja na poziomie 0.94 i 0.97 dla B i M. Pokrycie 0.98 dla raka łagodnego, 0.92 dla złośliwego. Jeśli chodzi o wynik F1 to uzyskano 0.96 dla B i 0.94 dla M. Dodatkowo macierz pomyłek wykazała 98 przypadków uzyskało status prawdziwie pozytywny a 65 – fałszywie negatywny w zbiorze testowym.

8.3 Drzewa decyzyjna

Algorytm drzewa decyzyjnego znany jest również pod bardziej nowoczesną nazwą CART, która oznacza drzewa klasyfikacji i regresji. Algorytm można zastosować do problemów z modelowaniem predykcyjnym klasyfikacji lub regresji. Reprezentacją algorytmu jest drzewo binarne. Każdy węzeł reprezentuje pojedynczą zmienną wejściową x i punkt podziału na tej zmiennej (przy założeniu, że zmienna jest liczbowa). Liście drzewa zawierają zmienną wyjściową y, która służy do przewidywania. Przewidywanie nowych danych polega na przejściu przez drzewo od głównego węzła. Uczenie się binarnego drzewa decyzyjnego jest procesem dzielenia przestrzeni wejściowej (podziały w drzewie) za pomocą algorytmu zachłannego wykorzystującego dane treningowe. Wszystkie wartości są zestawiane, a różne punkty podziału są wypróbowywane i testowane za pomocą funkcji kosztu, a następnie wybierany jest podział z najniższym kosztem. Wszystkie zmienne wejściowe i wszystkie możliwe punkty podziału są oceniane i

wybierane w zachłanny sposób (za każdym razem wybierany jest najlepszy punkt podziału).

Do klasyfikacji wykorzystywana jest funkcja kosztu Gini, która wskazuje, jak czyste są węzły liści (jak mieszane są dane treningowe przypisane do każdego węzła). Funkcja kosztu Gini ma postać:

$$G = \sum_{k=1}^{n} p_k * (1 - p_k)$$

gdzie G jest to koszt Gini we wszystkich klasach, pk jest liczbą instancji treningowych należących do klasy k. Węzeł, który ma wszystkie klasy tego samego typu (idealna czystość klas), będzie miał G = 0, podział 50-50 klas dla problemu klasyfikacji binarnej (najgorsza czystość) będzie miał wartość G = 0,5.

W procedurze rekurencyjnego dzielenia drzewa musi zostać podane kryterium stopu, aby algorytm wiedział, kiedy przestać dzielić, gdy przesuwa się w dół drzewa z danymi treningowymi. Najczęstszą procedurą zatrzymania jest użycie minimalnej liczby instancji treningowych przypisanych do każdego węzła liścia. Jeśli liczba jest mniejsza niż pewne minimum, podział nie jest akceptowany, a węzeł jest traktowany jako końcowy węzeł liścia. Liczba członków szkolenia jest dostosowywana do zestawu danych, np. 5 lub 10 [24].

Algorytm drzew decyzyjnych zaimplementowano zgodnie z poniższym kodem:

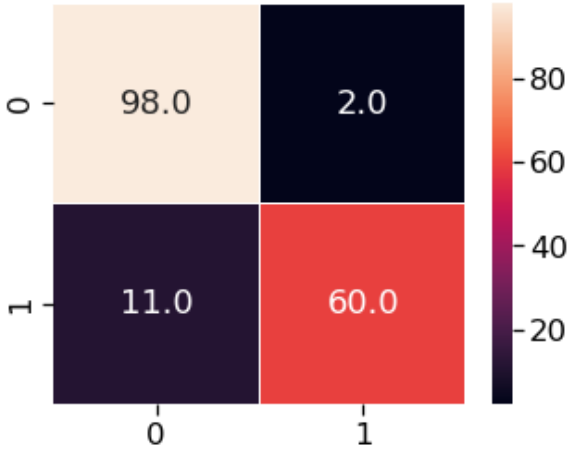
```
# Stworzenie modelu
model_decision_trees = DecisionTreeClassifier()

# Dopasowanie modelu
model_decision_trees.fit(X=x_train, y=y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

```
# Predykcja
prediction_decision_trees = model_decision_trees.predict(x_test)

# Ewaluacja modelu

# Raport klasyfikacyjny
results_decision_trees =
metrics.classification_report(y_true=y_test,
y_pred=prediction_decision_trees)
print(results_decision_trees)
```

```
# Macierz pomyłek
confusion_matrix_decision_trees=metrics.confusion_matrix(y_true=
y_test, y_pred=prediction_decision_trees)
f,ax = plt.subplots(figsize=(5, 4))
sns.heatmap(confusion_matrix_decision_trees, annot=True,
linewidths=.5, fmt= '.1f',ax=ax);
```

```
              precision    recall  f1-score   support

           B       0.90      0.98      0.94       100
           M       0.97      0.85      0.90        71

    accuracy                           0.92       171
   macro avg       0.93      0.91      0.92       171
weighted avg       0.93      0.92      0.92       171
```



Dokładność dla drzew decyzyjnych wyniosła 0.92. Dodatkowo uzyskano precyzje równą 0.97 (M) i 0.90 (B) oraz pokrycie wynoszące 0.98 dla B i 0.85 dla M. Natomiast wynik F1 0.94 dla raka łagodnego i 0.90 dla raka złośliwego. Macierz pomyłek wykazała, że prawdziwie pozytywnych wyników było 98, a fałszywie negatywnych 60.

8.4 K-NN, czyli K najbliższych sąsiadów

Algorytm k najbliższych sąsiadów zaliczany jest do grupy metod leniwych, czyli takich, które nie tworzą wewnętrznej reprezentacji danych uczących, ale szukają rozwiązania w momencie pojawienia się wzorca testowego. Przechowuje wszystkie wzorce uczące, względem których wyznacza odległość wzorca testowego.

Każdej danej jest przypisywany zestaw n cech, który jest umieszczany w n-wymiarowej przestrzeni. Jeśli chcemy przyporządkować daną do grupy to należy znaleźć k najbliższych sąsiadów w przestrzeni n-wymiarowej (najbliższych dla

wybranej metryki, np. Euklidesowej), a następnie wybrać grupę najbardziej liczną. Wzór na odległość Euklidesa:

$$\|x - x^i\| = \sum_{j=1}^{n} (x_j - x_j^i)^2$$

Gdzie x^i to zbiór parametrów $x^i = x_1^i, \dots, x_1^n$ definiujących obiekty (macierzy danych, wektory) zaś y^i jest wartością przewidywaną, indeksem lub nazwą klasy, do której obiekt x^i należy i którą razem z innymi obiektami tej klasy definiuje [25].

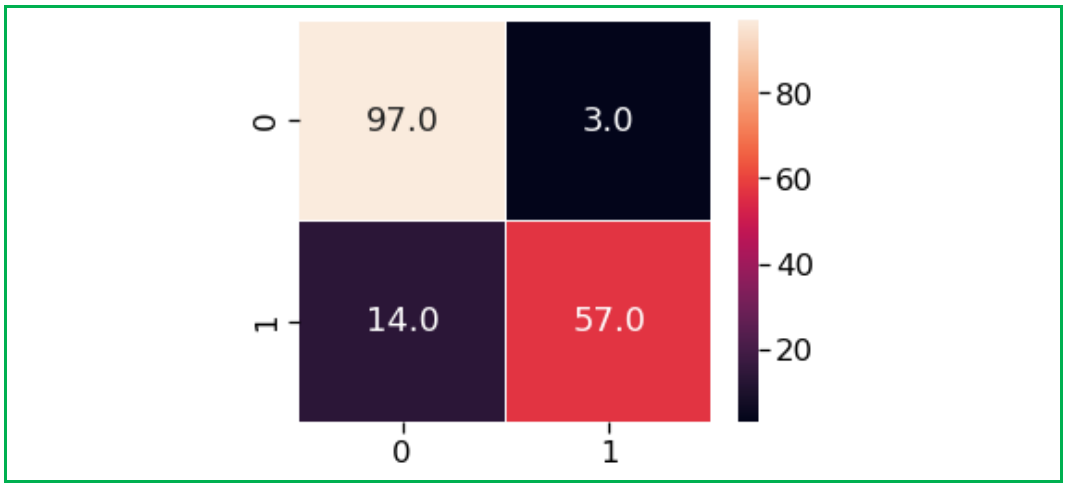
Algorytm K najbliższych sąsiadów został zaimplementowany w poniższy sposób:

```
# Stworzenie modelu
model_knn= KNeighborsClassifier(n_neighbors=10)

# Dopasowanie modelu
model_knn.fit(X=x_train,y=y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=10, p=2,
                     weights='uniform')
```

```
# Predykcja
predicion_knn = model_knn.predict(x_test)

# Ewaluacja modelu

# Raport klasyfikacyjny
results_knn=metrics.classification_report(y_true=y_test,
y_pred=predicion_knn)
print(results_knn)

# Macierz pomyłek
confusion_matrix_knn= metrics.confusion_matrix(y_true=y_test,
y_pred=predicion_knn)
f,ax = plt.subplots(figsize=(5, 4))
sns.heatmap(confusion_matrix_knn, annot=True, linewidths=.5,
fmt= '.1f',ax=ax)
```

```
              precision    recall  f1-score   support

           B       0.87      0.97      0.92       100
           M       0.95      0.80      0.87        71

    accuracy                           0.90       171
   macro avg       0.91      0.89      0.89       171
weighted avg       0.91      0.90      0.90       171
```

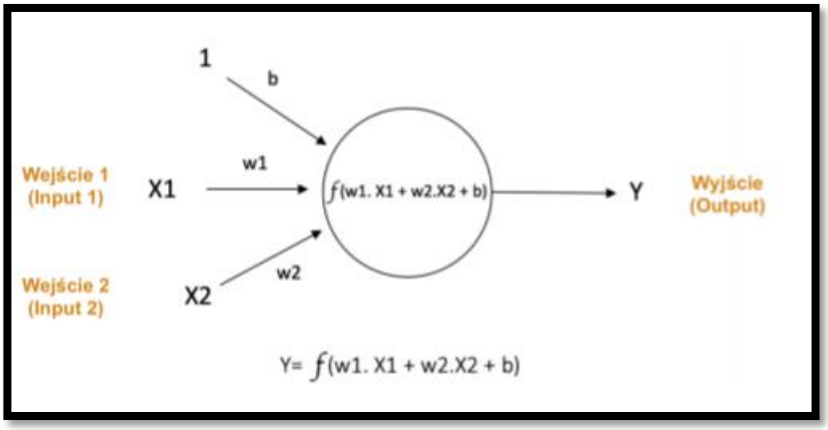


Model K najbliższych sąsiadów uzyskało dokładność równą 0.90, co czyni go najgorszą z pośród wszystkich pięciu modeli uczenia maszynowego. Precyzja dla M równa 0.95 i B – 0.87. Natomiast pokrycie dla B – 0.97, a dla M – 0.80. Wynik F1 uzyskał wynik 0.92 dla raka łagodnego i 0.87 dla złośliwego. Macierzy pomyłek dla k-NN pokazała, że model zdiagnozował poprawnie pozytywnie 97 przypadków i 57 fałszywie negatywnie.

8.5 Sztuczne sieci neuronowe

Sztuczna sieć neuronowa to model obliczeniowy zainspirowany biologiczną siecią neuronów w ludzkim mózgu. u. Sieć neuronową można opisać jako ukierunkowany wykres, którego węzły odpowiadają neuronom, a każda (skierowana) krawędź na wykresie łączy wyjście niektórych neuronów z wejściem innego neuronu (połączenia między nimi).

Podstawową jednostką obliczeniową tego modelu jest neuron zwany perceptronem. Na początku cechy wejściowe przyjmuje wektor $\vec{x}$ i neuron otrzymuje jako dane wejściowe ważoną sumę wyników neuronów połączonych jego przychodzącymi krawędziami. Perceptronem zwraca jedną binarną wartość jako output.



Rys. 11. Podstawowa jednostka sieci neuronowej, czyli neuron.

Każdemu wejściu x_i jest przypisana waga w_i, która jest wyznaczona na podstawie jego znaczenia w porównaniu ze wszystkimi danymi wejściowymi. W węźle wykonywana jest nieliniowa funkcja f (ważona suma wejść) nazywaną funkcją aktywacji. Natomiast wyjście neuronu wyliczane jest ze wzoru:

$$Y = f(w_1x_1 + w_2x_2 + \cdots + b) = f(\sum_{i=1}^{n}, w_ix_i + b)$$

Jeśli chodzi o funkcje aktywacji to mają różne postaci.

- Funkcja sigmoidalna, która pobiera dane wejściowe o wartościach rzeczywistych, zwraca wartości w przedziale od 0 do 1:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- Tanges hiperboliczny (tanh), która pobiera dane wejściowe o wartościach rzeczywistych, zwraca wartości w przedziale od -1 do 1:

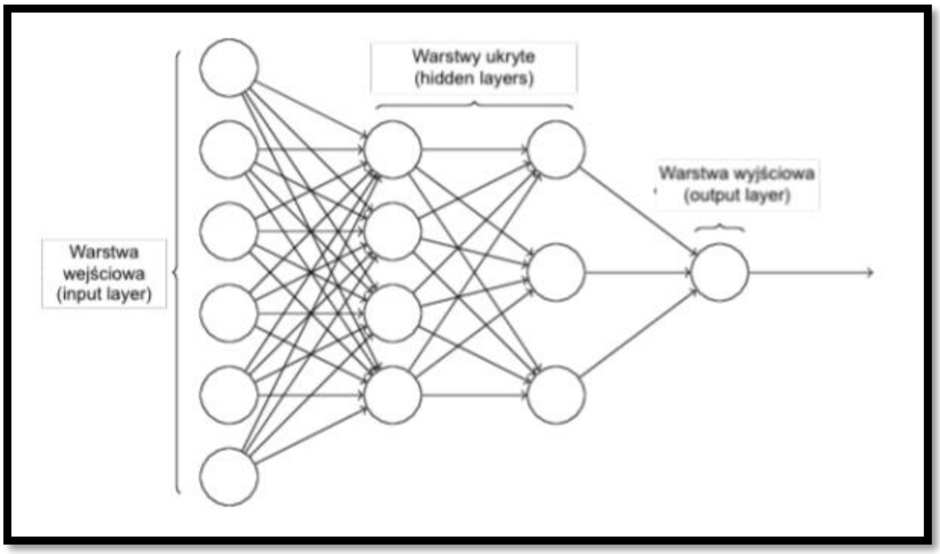
$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

- ReLU, która pobiera dane wejściowe o wartościach rzeczywistych, wartości ujemna zastępuje zerem:

$$f(x) = \max(0, x)$$

W sieci neuronowej, warstwą wejściową nazywamy skrajnie lewą warstwę. Umieszczone w niej neurony to neurony wejściowe, które dostarczają informacje ze świata zewnętrznego do ukrytych węzłów. Warstwa wyjściowa (skrajna - po prawej) zawiera jeden lub więcej neuronów wyjściowych, które są odpowiedzialne za obliczenia i przesyłanie informacji z sieci do świata zewnętrznego. Warstwy środkowe zwane są warstwami ukrytymi, ponieważ nie mają bezpośredniego

połączenia ze światem zewnętrznym. Przykładowa sieć neuronowa przedstawiono poniżej, która ma 4 warstwy: wejściową, wyjściową i 2 ukryte.



Rys. 12. Przykładowa wielowarstwowa sieć neuronowa.

Poniżej przedstawiono kod stworzonego modelu sieci neuronowych dla zbioru *Breast Cancer Wisconsin (Diagnostic) Data Set:*

```
np.random.seed(0)

# Stworzenie modelu składający się z wielu wartsw
model_neural_network = Sequential()
model_neural_network.add(Dense(units=160,
kernel_initializer='uniform', activation='relu', input_dim=30))

# Dodanie warstw ukrytych
model_neural_network.add(Dense(units=160,
kernel_initializer='uniform', activation='relu'))
model_neural_network.add(Dense(units=160,
kernel_initializer='uniform', activation='relu'))
model_neural_network.add(Dense(units=160,
kernel_initializer='uniform', activation='relu'))
model_neural_network.add(Dense(units=160,
kernel_initializer='uniform', activation='relu'))
model_neural_network.add(Dense(units=160,
kernel_initializer='uniform', activation='relu'))
model_neural_network.add(Dense(units=160,
kernel_initializer='uniform', activation='relu'))
model_neural_network.add(Dense(units=160,
kernel_initializer='uniform', activation='relu'))
model_neural_network.add(Dense(units=160,
kernel_initializer='uniform', activation='relu'))
model_neural_network.add(Dense(units=160,
kernel_initializer='uniform', activation='relu'))
model_neural_network.add(Dense(units=160,
kernel_initializer='uniform', activation='relu'))
```

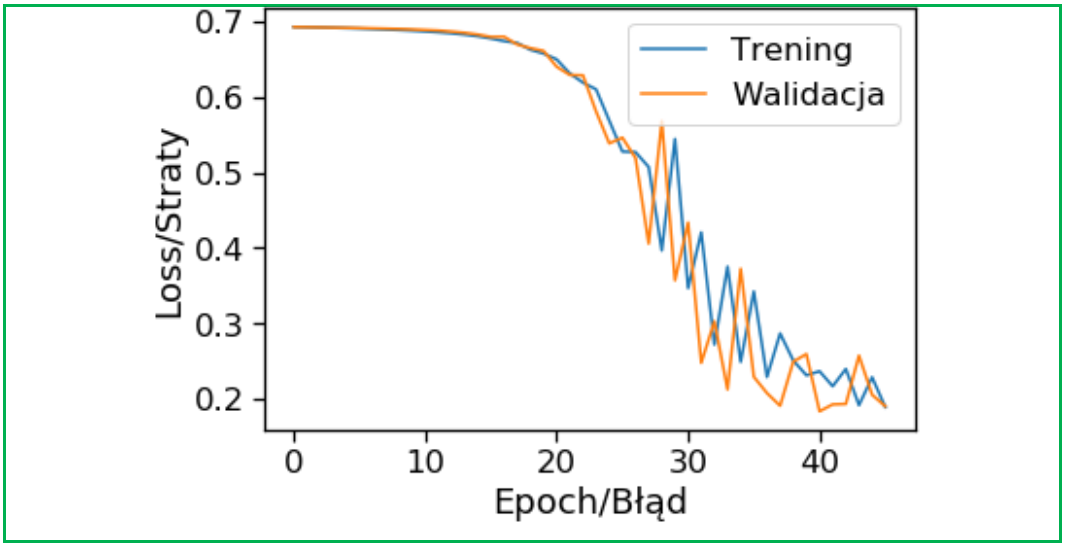
```
# Dodanie warstwy wyściowej
model_neural_network.add(Dense(units=1,
kernel_initializer='uniform', activation='sigmoid'))

# Użycie loss function: 'Binary_crossentropy'
model_neural_network.compile(loss='binary_crossentropy',
optimizer='adam', metrics=['accuracy'])

#Dostosowanie modelu przy uzyciu poprzedniego optymalizatora
earlystop=callbacks.EarlyStopping(monitor='val_loss',
min_delta=0, patience=5, mode='auto',verbose=1)
history=model_neural_network.fit(x_train,
pd.get_dummies(y_train,drop_first=True)['M'].values,
validation_split=0.2, epochs=500, batch_size=5000, verbose=0,
callbacks=[earlystop])
```

```
Epoch 00046: early stopping
```

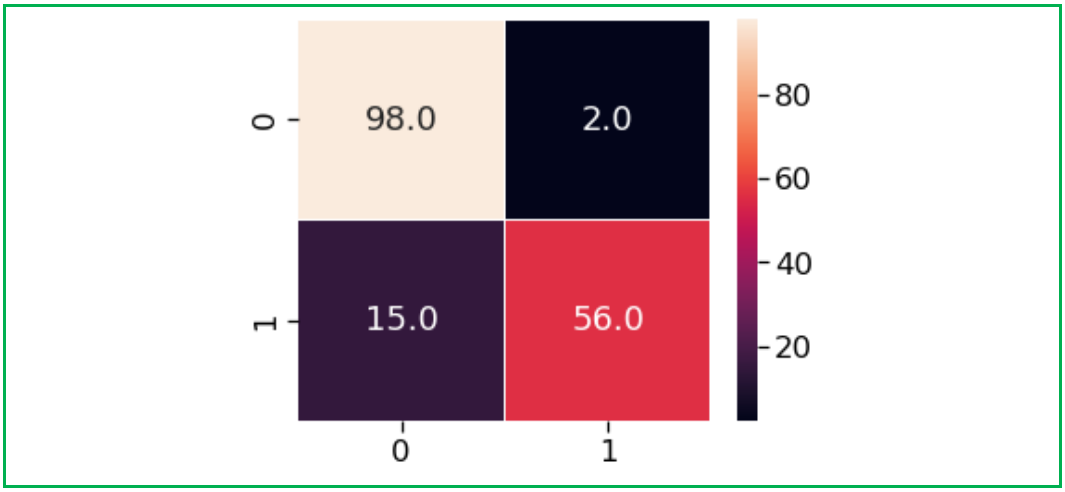
```
# Predykcja klas
predicion_neural_network =
model_neural_network.predict_classes(x_test, batch_size=32)

# Raport klasyfikacyjny
results_neural_network =
metrics.classification_report(y_true=pd.get_dummies(y_test,
drop_first=True), y_pred=predicion_neural_network)
print(results_neural_network)
```

```
              precision    recall  f1-score   support

           0       0.87      0.98      0.92       100
           1       0.97      0.79      0.87        71

    accuracy                           0.90       171
   macro avg       0.92      0.88      0.89       171
weighted avg       0.91      0.90      0.90       171
```

```
# Zależność funkcji strat od epoki
plt.plot(history.history['loss']);
plt.plot(history.history['val_loss']);
plt.title('Straty w modelu')
plt.ylabel('Loss/Straty')
plt.xlabel('Epoch/Błąd')
plt.legend(['Trening','Walidacja'], loc='upper right');
plt.show()
```




```
# Macierz pomyłek
confusion_matrix_neural_network=metrics.confusion_matrix(y_true=
pd.get_dummies(y_test,drop_first=True),
y_pred=predicion_neural_network)
f,ax = plt.subplots(figsize=(5, 4))
sns.heatmap(confusion_matrix_neural_network, annot=True,
linewidths=.5, fmt= '.1f',ax=ax)
```



Na podstawie stworzonej sztucznej sieci neuronowej uzyskano model, który po wytrenowaniu uzyskał dokładność równą 0.90. Dodatkowo precyzja wyniosła 0.87 dla raka łagodnego (0) oraz 0.97 dla raka złośliwego (1). Natomiast pokrycie dla tego modelu dla B (0) wyniosło 0.92 i dla M (1) – 0.79. Kolejno wynik F1 uzyskano na poziomie 0.92 dla nowotworu łagodnego oraz 0.87 dla złośliwego. Za pomocą macierzy pomyłek uzyskano 98 poprawnie pozytywnych wyników oraz fałszywie negatywnych 56. Model sieci neuronowych wypada średnio w porównaniu z resztą modeli ML.

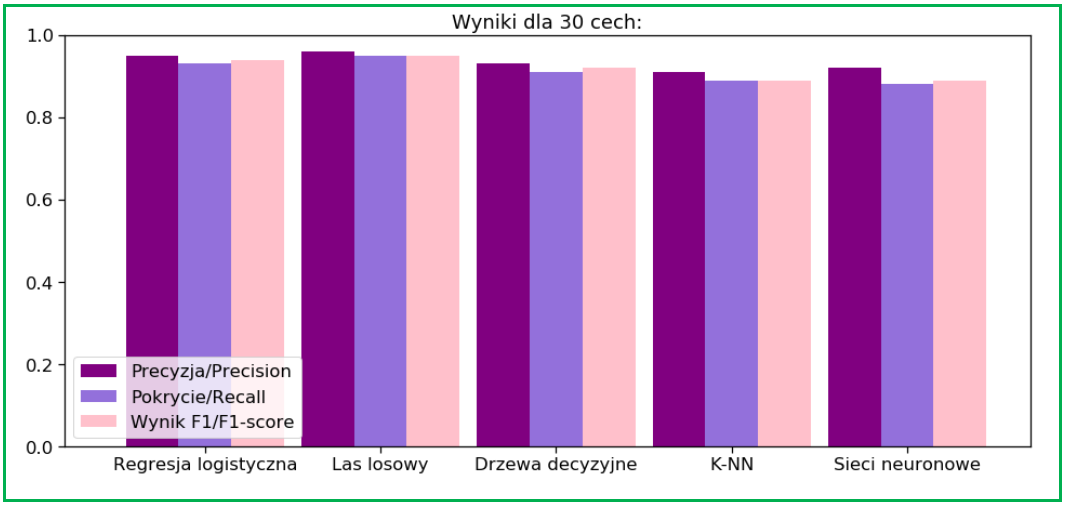
8.6 Porównanie modeli uczenia maszynowego

W rozdziale zestawiono ze sobą wszystkie pięć modeli Machine Learning oraz porównano wyniki precyzji, pokrycia, wyniku F1 oraz dokładność.

Wykreślono wykres słupkowy dla regresji logistycznej, lasu losowego, drzew decyzyjnych, k-NN i sieci neuronowych w celu porównania ewaluacji modelu. Kod przedstawiono poniżej:

```
# Definicja funkcji, która zwraca sumę (ostatni wiersz) raportu
klasyfikacji w tablicy
# Kolejność: precyzja/precision, pokrycia/recall, wynik F1/wynik
F1, wsparcie/support

def results_classification_report(cr):
    total=[]
    lines = cr.split('\n')
    total_aux=lines[6].split()
    for i in range(2,5):
        total.append(float(total_aux[i]))
    return total

# Zebranie raportów klasyfikacyjnych
names=['Regresja logistyczna','Las losowy','Drzewa
decyzyjne','K-NN','Sieci neuronowe']
model_results=[]
model_results.append(results_logistic_regression)
model_results.append(results_random_forest)
model_results.append(results_decision_trees)
model_results.append(results_knn)
model_results.append(results_neural_network)

total_results =[]
total_precision = []
total_recall = []
total_F1 = []

for i in range(len(model_results)):
    total_results.append(results_classification_report
     (model_results[i]))

for i in range(len(total_results)):
    total_precision.append(total_results[i][0])
    total_recall.append(total_results[i][1])
    total_F1.append(total_results[i][2])


# Podsumowanie - Raport klasyfikacyjny dla 5 modeli
index = np.arange(5)

# Ustawienie bar na osi X
barWidth=0.3
r1 = np.arange(len(total_precision))
r2 = [x + barWidth for x in r1]
r3 = [x + barWidth for x in r2]
```

```
plt.subplots(figsize=(16, 7))
plt.bar(r1,total_precision,width=barWidth,
label='Precyzja/Precision', color="purple")
plt.bar(r2,total_recall,width=barWidth, label='Pokrycie/Recall',
color="mediumpurple")
plt.bar(r3,total_F1,width=barWidth,  label='Wynik F1/F1-score',
color="pink")
plt.axis([-0.5,5,0, 1])
plt.legend(loc='lower left')
plt.title('Wyniki dla 30 cech:')

# Dodanie etykiet do wykresu
plt.xticks([r + barWidth for r in range(len(total_precision))],
names)
```



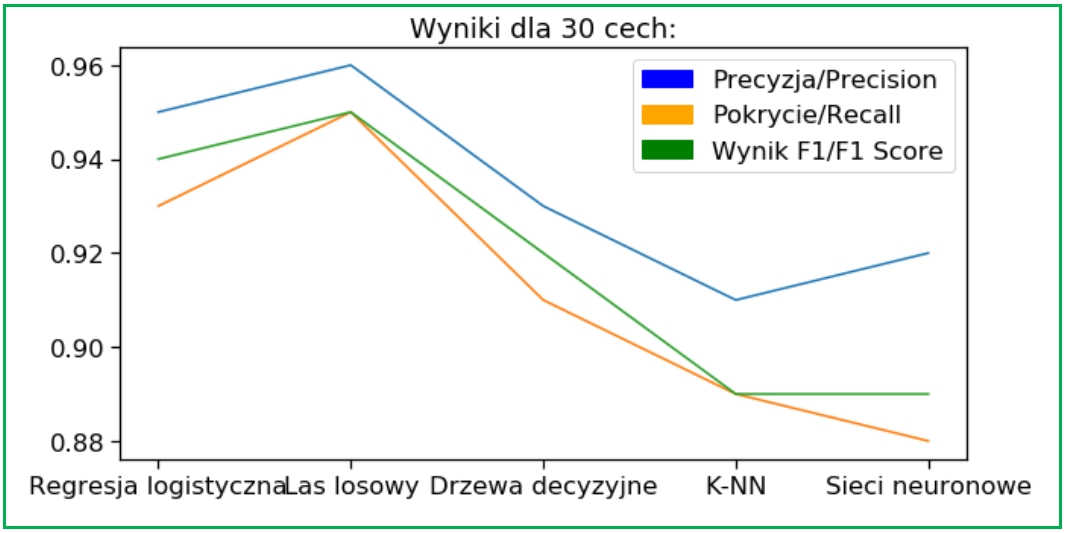
Możemy zaobserwować, że wyniki dla precyzji, pokrycia i wyniku F1 były w zakresie 0,9-1. Precyzja dla wszystkich modeli była na bardzo wysokim poziomie, co świadczy o tym, że wszystkie modele są w stanie przewidzieć poprawnie wyniki na podstawie danych. Inaczej mówią proponowana predykcja dla wszystkich modeli mieści się w przedziale 0.9-1. Najwyższą precyzje uzyskane dla modelu lasu losowego: dla przypadków B (rak łagodny) równą 0.94 i dla M (rak złośliwy) równą 0.97.

Pokrycie sprawdza prawdopodobieństwo, że model przewidzi daną wartość zgodnie z prawdą. Wyniki uzyskane mieściły się w przedziale ok. 0.88-95. Najwyższy wynik pokrycia moża zaobserwować również dla lasu losowego, ponieważ wynik jest na poziomie 0.98 dla B i 0.92 dla M.

Najlepszy wynik F1, czyli średniej harmonicznej precyzji i pokrycia, uzyskano również dla lasu losowego, które wynoszą 0.96 dla B i 0.94 dla M.

Po wstępnej analizie wyników precyzji, pokrycia i wyniku F1, zestawiono je na wykresie liniowym w węższym zakresie 0.88 do 0.96, by lepiej zaobserwować zmiany. Implementowany kod przedstawiono poniżej:

```
# Wykres przedstawiający precyzje, odwołanie i wynik F1 dla 30
cech dla 5 modeli
legend=[]
legend.append(mpatches.Patch(color='blue',
label='Precyzja/Precision'))
legend.append(mpatches.Patch(color='orange',
label='Pokrycie/Recall'))
legend.append(mpatches.Patch(color='green', label='Wynik F1/F1
Score'))

plt.subplots(figsize=(10, 5))
plt.plot(names,total_results)
plt.legend(handles=legend)
plt.title('Wyniki dla 30 cech:')
```

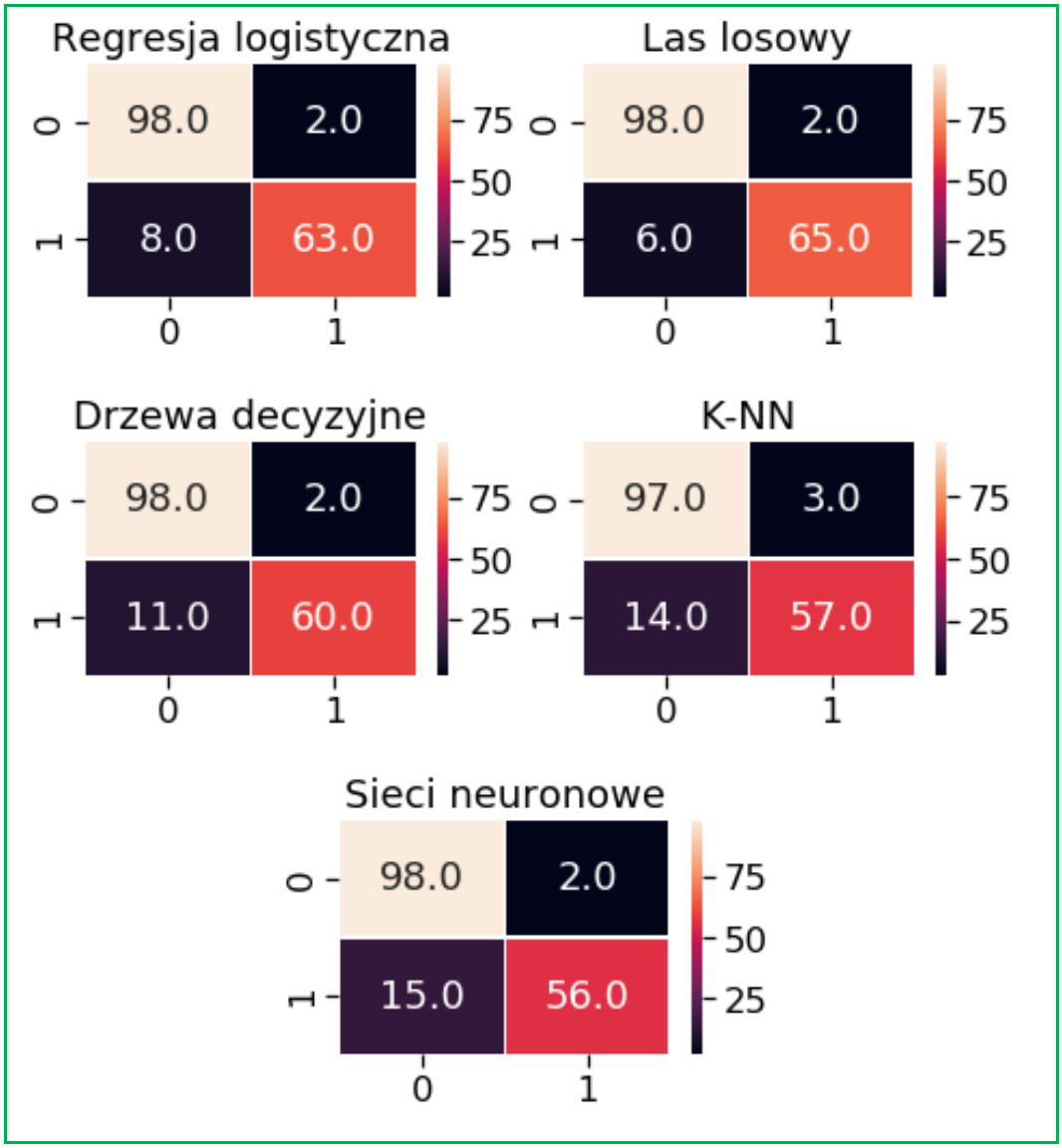


Wykres potwierdził, że najlepsze wyniki uzyskano dla lasu losowego, a najgorsze dla sztucznych sieci neuronowych. Na drugim miejscu znalazła się regresja logistyczna z wynikiem: precyzja dla B – 0.92, M – 0.97, pokrycie dla B – 0.98, M – 0.98 oraz wynik F1 dla B – 0.95, M-0.93. Na trzecim miejscu znalazły się drzewa decyzyjne, a na czwartym – k najbliższych sąsiadów.

Następnie przygotowano zestawienie macierzy pomyłek dla pięciu modelu ML i poniżej przedstawiono kod dla programu:

```
# Zestawienie macierzy pomyłek dla wszystkich modeli
f,ax = plt.subplots(figsize=(20, 2))
plt.subplot(1,5,1)
plt.title('Regresja logistyczna')
sns.heatmap(confusion_matrix_logistic_regression, annot=True,
linewidths=.5, fmt= '.1f')

plt.subplot(1,5,2)
plt.title('Las losowy')
sns.heatmap(confusion_matrix_random_forest, annot=True,
linewidths=.5, fmt= '.1f')
```

```
plt.subplot(1,5,3)
plt.title('Drzewa decyzyjne')
sns.heatmap(confusion_matrix_decision_trees, annot=True,
linewidths=.5, fmt= '.1f')

plt.subplot(1,5,4)
plt.title('K-NN')
sns.heatmap(confusion_matrix_knn, annot=True, linewidths=.5,
fmt= '.1f')

plt.subplot(1,5,5)
plt.title('Sieci neuronowe')
sns.heatmap(confusion_matrix_neural_network, annot=True,
linewidths=.5, fmt= '.1f')
```



Każdy model z wyjątkiem k-NN uznał 98 przypadków za TP (prawdziwie pozytywny), czyli 98 analiz było prawdziwe analizowanych dla wartości rzeczywistych i przewidywanych. Dodatkowo 2 przypadki trafiły do FP (fałszywie

prawdziwy), czyli zostały zanalizowane poprawnie dla wartości przewidzianych, ale fałszywie dla rzeczywistych. W dwóch pozostałych grupach FP (fałszywie negatywny) i F (negatywny) wyniki dla różnych modeli były bardziej rozbieżne. Najwięcej wyników negatywnych (65) uzyskano dla lasu losowego i 6 przypadków dla FP. Najmniej jednak dla sztucznych sieci neuronowych. W przypadku macierzy pomyłek las losowy znowu wypadł najlepiej bo zdiagnozował podobnie wyniki prawdziwie pozytywne z małą liczbą fałszywie prawdziwych. Dodatkowo zdiagnozował najwięcej negatywnych wyników z małą liczbą wyników prawdziwie negatywnych. Oznacza to, że model lasu losowego najlepiej poradził sobie z wykryciem błędów w bazie i poprawną identyfikacją czy dany wynik jest zgodny z rzeczywistością czy nie.

9 Podsumowanie

W poruszającej zagadnienia *Machine Learning* udało się wykonać pierwszy cel, czyli podstawową analizę zbioru *Breast Cancer Wisconsin (Diagnostic) Data Set*. Dowiodła ona, że w zbiorze danych znajduje się 568 rekordów diagnozowanych przypadków raka piersi z czego 357 to nowotwory łagodne, a 211 złośliwe. Wartości cech opisujące nowotwory są bardzo różne o bardzo szerokich zakresach. Na tym etapie zaobserwowano wzrost wszystkich wartości wraz ze zwiększającymi się cechami opisującymi wielkość raka.

Kolejnym etap pracy była wizualizacja danych za pomocą macierzy korelacji, wykresu rozrzutu, wykresu ramka-wąsy oraz wykresu skrzypcowego. Za pomocą macierzy korelacji i wykresu rozrzutu udało się znaleźć kilka bardzo silnych korelacji. Na podstawie nich udało się wywnioskować, że wraz z namnażaniem komórek rakowych i wzrostem ich ilości zwiększa się obszar jaki zajmują jak i promień. Dodatkowo ze wzrostem wartości promienia, obszaru i obwodu guza rośle liczba jego punktów wklęsłości na powierzchni. Na podstawie sporządzonych wykresów ramka-wąsy można zauważyć, że wartości cech dla raka złośliwego mają większe wartości od raka łagodnego. Złośliwy nowotwór piersi posiada większy promień, obwód i zajmowany obszar oraz bardziej nierównomierną powierzchnie o mniejszej symetrii od raka łagodnego. Można podejrzewać zatem, że komórki raka M mogą namnażać się szybciej od komórek B oraz powodować gorsze szkody w ludzkim organizmie oraz większą ilość przypadków śmiertelnych. W przypadku wykresów skrzypcowych zauważono, że przypadki najgorsze dla raka złośliwego mają większe wartości cech (większe skrzypce = większą gęstość prawdopodobieństwa) od najgorszych przypadków raka łagodnego, przez co nowotwór złośliwy może powodować większą śmiertelność.

Następnym krokiem oraz głównym celem pracy było stworzenie pięciu klasyfikatorów z wykorzystaniem narzędzi uczenia maszynowego, których zadaniem było zdiagnozowanie raka piersi zbioru testowego. Zaimplementowano model regresji logistycznej, lasu losowego, drzew decyzyjnych, k najbliższych sąsiadów oraz sztucznych sieci neuronowych. Za pomocą parametrów tj. macierz pomyłek, precyzja, pokrycie, wynik F1 oraz dokładność, oceniono każdy z modeli pod kątem radzenia sobie z postawieniem odpowiedniej diagnozy. Wszystkie modele uzyskały zadawalające rezultaty, czyli dokładność na poziomie 0.90. Najlepszym modelem okazał się klasyfikator lasu losowego, który miał najwyższy wynik dokładności, precyzji, pokrycia, wynik F1 (0.92-0.98). Dodatkowo w macierzy pomyłek dla 98 przypadków otrzymał wynik prawdziwie pozytywny a dla 65 fałszywie negatywny, co pokazuje, że las losowy najbardziej umiał poprawnie zdiagnozował przypadki i odseparować wyniki niepoprawne. Najmniej ze wszystkich uzyskał wyników fałszywie pozytywnych, bo tylko 2 oraz

pozytywnie negatywnych – 6. Natomiast najmniej sprawdził się model k najbliższych sąsiadów, który uzyskał najgorsze wyniki podczas ewaluacji modelu.

Podsumowując, przeprowadzanie podobnych analiz *Machine Learning* daje niesamowitą nadzieję na rozwój diagnostyki raka piersi oraz innych ciężkich chorób. Wyniki uzyskane w tej pracy oraz publikacje naukowe z całego świata pokazują, że to tylko kwestia czasu kiedy będzie to codzienność wspomagająca prace personelu medycznego zwłaszcza, że w niektórych przypadkach modele ML radzą sobie lepiej w rozwiązywaniu problemów od ludzi i specjalistów z różnych dziedzin.

10 Bibliografia

[1] M. K. Jacek Jassem, „Breast cancer," *Oncology in Clinical Practice,* pp. 171-215, 3 marzec 2018.

[2] M. V. Jaak Ph. Janssens, „Breast cancer: a life-time disease. Direct and indirect age-related lifestyle risk factors.," *Journal of Oncology,* pp. 159-167, 2009.

[3] A. Bloomer, „Artificial intelligence effective in breast cancer diagnosis," Styczeń 2020. [Online]. Available: https://www.gmjournal.co.uk/artificial-intelligence-effective-in-breast-cancer-diagnosis.

[4] „The Guardian," Styczeń 2020. [Online]. Available: https://www.theguardian.com/society/2020/jan/01/ai-system-outperforms-experts-in-spotting-breast-cancer.

[5] „Pandas Dokumentation," [Online]. Available: https://pandas.pydata.org/docs/user_guide/10min.html.

[6] „NumPy," [Online]. Available: https://numpy.org/.

[7] „Matplotlib," [Online]. Available: https://matplotlib.org/.

[8] „Seaborn," [Online]. Available: https://seaborn.pydata.org/.

[9] „Scikit-learn," [Online]. Available: https://scikit-learn.org/stable/.

[10] „Keras," [Online]. Available: https://keras.io/.

[11] W. N. S. O. L. M. William H. Wolberg, „Breast Cancer Wisconsin (Diagnostic) Data Set," UCI Machine Learning Repository, [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic).

[12] J. Kornafel, „Rak piersi," *Cemtrum Medyczne Kształcenia Podyplomowego w Warszawie,* 2011.

[13] „Macierz korelacji," [Online]. Available: https://pl.wikipedia.org/wiki/Macierz_korelacji.

[14] A. G. Wojciech Szpara, „Wykres pudełkowy,” [Online]. Available: https://mfiles.pl/pl/index.php/Wykres_pude%C5%82kowy.

[15] M. Mamczur, „Wykres skrzypcowy,” [Online]. Available: https://miroslawmamczur.pl/005-wykres-skrzypcowy-violin-plot/.

[16] M. Mamczur, „Na czym polega analiza składowych głównych (PCA)?,” Mirosław Mamczur, 16 Październik 2019. [Online]. Available: https://miroslawmamczur.pl/na-czym-polega-analiza-skladowych-glownych-pca/.

[17] K. W.J., Principles of Multivariate Analysis: A User’s Perspective, Oxford University Press, 2000.

[18] M. Mamczur, „Jak działa redukcja wymiarów t-SNE?,” [Online]. Available: https://miroslawmamczur.pl/jak-dziala-metoda-redukcji-wymiarow-t-sne/.

[19] „Regresja logistyczna,” Statystyka od A do Z, [Online]. Available: https://www.statystyka.az.pl/regresja-logistyczna.php.

[20] S. Narkhede, „Understanding confusion matrix,” *Towards Data Science,* 2018.

[21] P. M. l. w. c. 3. –. t. m. Owsianik, „ Machine learning workflow cz 3 – testowanie modelu.,” *Thinking in code*, 2018.

[22] D. M. Powers, „Evaluation: From Precision, Recall and F-Score to ROC, Informedness, Markedness & Correlation,” *Journal of Machine Learning Technologies,* p. 2 (1): 37–63, 2011.

[23] Las losowy, [Online]. Available: https://pl.wikipedia.org/wiki/Las_losowy.

[24] J. Brownlee, „Master Machine Learning Algorithms. Discover How They Work and Implement Them From Scratch.,” *Machine Learning Mastery*, 2016.

[25] A. Horzyk, Metody inteligencji obliczeniowej, 2015.