Analiza wpływu zastosowania wybranych technik przygotowania danych do analizy, na jakość analizy danych

Dariusz Litwiński

12 czerwca 2023

Spis treści

Rozdział 1

Problem Analizy Danych

1.1 Analiza Danych

Analiza danych to proces, w którym przekształcamy surowe dane w wiedzę i wnioski, dzięki którym jesteśmy w stanie podejmować lepsze decyzje [4]. Wewnątrz tego procesu można wyróżnić następujące fazy:

1.1.1 Pozyskiwanie: gromadzenie danych

Zanim analiza danych będzie możliwa należy pozyskać dane, jest to istotna cześć procesu, ponieważ to od niej zależy to, jak trafne wnioski będziemy mogli wysnuć na późniejszych etapach. Bardzo ważne jest to, jak dużą ilość danych uda nam się zebrać, a także jak dokładne one będą. Możemy pozyskiwać dane z różnorakich źródeł, na przykład:

- Zapisywać wartości zbierane przez czujniki [5]
- Korzystać z ankiet zebranych wśród danej populacji [7]
- Zbierać dane o zachowaniach użytkowników w trakcie korzystania ze strony internetowej [8]
- Wykorzystywać dane statystyczne/historyczne [6]

Jesteśmy w tej kwestii ograniczeni jedynie przez dziedzinę, w jakiej przeprowadzamy analizę danych Istotną kwestią jest również to, w jakim formacie są przechowywane dane. Może okazać się, że pierwszym etapem przetwarzania danych będzie odpowiednie ich sformatowanie, bądź nawet ich cyfryzacja, jeśli były by przechowywane w formie fizycznej, na przykład jako papierowe archiwa

1.1.2 Przygotowanie: przetwarzanie danych

Aby móc w pełni korzystać ze zgromadzonych danych, należy je przygotować, odpowiednio sformatować. Poza przygotowaniem odpowiedniego formatu da-

nych, możemy wyróżnić następujące sposoby na przygotowanie danych do analizy:

- Wypełnienie brakujących wartości
- Standaryzacja danych liczbowych
- Kodowanie wartości kategorycznych jako liczbowe

Dzięki wykorzystaniu powyższych sposobów możemy znacznie zwiększyć jakość analizy danych, a co za tym idzie wnioski do których dojdziemy w trakcie procesu będą trafniejsze i przyniosą lepsze rezultaty

1.1.3 Analiza: modelowanie danych

Mając do dyspozycji przygotowane dane możemy przejść do ich właściwej analizy, na podstawie której tworzymy modele, klasyfikatory [15], systemy rekomendacji [16], dokonujemy klasteryzacji [17]. Najczęściej nie tworzymy ich od zera, a wspomagamy się dostępnymi bibliotekami, które zawierają najpopularniejsze algorytmy. Zdarza się, że nie jesteśmy zadowoleni z wyników, jakie przynosi wykorzystanie stworzonych modeli, bądź chcielibyśmy dokonać dalszej optymalizacji czasowej, w takim przypadku możemy rozważyć dodatkowe przygotowanie danych, aby uzyskać pożądany efekt. Po dobrze przeprowadzonej analizie, jesteśmy w stanie wykorzystać stworzone na tym etapie narzędzia do rozwiązywania rzeczywistych problemów.

1.1.4 Działanie: podejmowanie decyzji

Mając gotowe narzędzia będące wynikiem modelowania danych, możemy je wykorzystać aby podjąć konkretne decyzje w prawdziwym świecie: Wykorzystać stworzony klasyfikator w diagnostyce chorób, wdrożyć system rekomendacji na naszej stronie internetowej, wykorzystać stworzone klastry danych do kategoryzacji klientów. Każdy z przedstawionych problemów wymaga eksperckiej wiedzy oraz ogromnego doświadczenia, a dzięki analizie danych możemy znacznie usprawnić ich rozwiązywanie.

1.2 Jakość Danych

Jakość danych jest oceniana na podstawie wielu kryteriów, zależnych od źródła informacji[24]:

- Kompletność - ilość danych która jest kompletna bądź zdatna do użycia. Jeśli duża część danych jest niekompletna, może to prowadzić do stronniczej bądź nawet omylnej analizy.
- Unikalność - jaka część z zestawu danych się powtarza, dla przykładu zestaw danych zawierających dane o klientach powinien każdemu z nich przypisać unikalny numer identyfikacyjny

- Ważność - To kryterium mówi o tym czy zebrane dane są w odpowiednim formacie
- Aktualność - W zależności od dziedziny którą się zajmujemy, dane mogą tracić na aktualności wraz z upływem czasu, przez na przykład postęp w danej dziedzinie bądź zmieniające się warunki
- Dokładność - poprawność danych w oparciu o ustalone "źródło prawdy". Jako, że może istnieć wiele źródeł tych samych danych, należy ustalić nadrzędne źródło danych, pozostałe zaś mogą potwierdzać dokładność tego pierwszego.
- Stałość - kryterium służące do porównywania danych z dwóch zestawów danych. Używanie różnych źródeł do szukania stałych trendów w danych. Dzięki temu wnioski pojawiające się w trakcie analizy mogą być traktowane z większym zaufaniem, jako że tworzone są na podstawie wielu niezależnych zestawów danych.
- Dopasowanie do celu - to kryterium pozwala nam zapewnić fakt, że dane które są zbierane posłużą nam do rozwiązania problemu, jaki przed nami stoi

1.3 Problemy z Danymi

Bardzo często zebrane dane nie nadają się bezpośrednio do pracy z nimi. Należy najpierw wykonać szereg operacji aby pozbyć się następujących problemów:

1.3.1 Brakujące wartości

W danych mogą występować brakujące wartości, na przykład czujniki mogą różnić się między sobą ilością pobieranych parametrów, ankietowani mogą pozostawić niektóre pytania bez odpowiedzi. Brakujące wartości stanowią poważny problem, ponieważ model nie potrafi ich jednoznacznie zinterpretować, dlatego w trakcie przygotowania danych musimy podjąć decyzję, czy usunąć rekordy z brakującymi wartościami, przez co możemy znacznie zmniejszyć liczebność zbioru danych. Alternatywnym podejściem jest wypełnienie brakujących wartości. W miejsce braku może być wstawiona średnia, minimum, maksimum, lub też inna arbitralnie wybrana wartość. Pozornie wartości wstawiane w puste miejsca są kompletnie arbitralne, jednak bardzo często takie podejście skutkuje najlepszymi rezultatami, pod warunkiem że dobierzemy odpowiednią wartość do wstawiania. [9]

1.3.2 Wartości odstające

W niektórych przypadkach w danych mogą pojawić się takie wartości, które wyraźnie odstają od reszty i nie wnoszą sobą zbyt wiele informacji w kontekście analizy danych. Co więcej, mogą one zaciemniać pozostałą część danych,

maskując trendy bądź prowadząc do błędnych wniosków. Dlatego najlepszym podejściem jest wykrywanie oraz usuwanie wartości, które możemy uznać za odstające. Istnieją algorytmy pozwalające nam na odrzucenie wartości odstających. [12]

1.3.3 Kolumny kategoryczne

Wiele z modeli może pracować jedynie na wartościach liczbowych, podczas kiedy w zbiorach danych możemy znaleźć nie tylko takie wartości, ale również kategoryczne. Rezygnując z analizy tych danych tracilibyśmy wiedzę, jaką można z nich pozyskać. Nie jest to jednak konieczne, gdyż istnieją sposoby, aby zamienić te dane na postać liczbową za pomocą kodowania [14]

1.4 Cel pracy

Celem pracy jest sprawdzenie, jak poszczególne metody przygotowania danych wpływają na jakość analizy danych, konkretnie modelowania klasyfikatorów. Przeprowadzono eksperymenty z użyciem wielu metod przygotowania danych, a następnie porównano trafność klasyfikacji względem klasyfikatora bez przygotowania danych. Za brak przygotowania danych uznaje się usunięcie wszystkich rekordów z brakującymi wartościami oraz wszystkich kolumn nieliczbowych. Rozpoczynając prace postawiono hipotezę, że najlepszym sposobem na przygotowanie danych jest dogłębne ich zrozumienie, a następnie dostosowanie do nich użytych metod, jak i rówież fakt, że jakiekoliwek przygotowanie danych powinno wpłynąć pozytywnie na jakość analizy danych.

Rozdział 2

Metody Przygotowania Danych

Istnieje kilka najczęściej używanych metod przygotowania danych, które dzielą się na następujące grupy:

2.1 Uzupełnianie brakujących wartości

Najczęściej brakujące wartości w zbiorach danych uzupełnia się średnią lub najczęściej występującą wartością, jednak może zdarzyć się tak, że najlepszym rozwiązaniem jest uzupełnienie braków minimum, maksimum, zerem bądź inną arbitralnie wybraną wartością

2.2 Wykrywanie wartości odstających

Do wykrywania wartości odstających możemy wykorzystać manualne metody, ale również i algorytmy, które wykryją te wartości za nas. Do manualnych metod możemy zaliczyć: Wykrywanie za pomocą rozkładu normalnego, Z-score, IQR, wykrywanie za pomocą percentyli. Natomiast spośród automatycznych metod mamy do dyspozycji między innymi Las Izolacji lub Local Outlier Factor

2.3 Kodowanie wartości kategorycznych

Jeżeli znamy zależności między klasami i możemy je uporządkować, wtedy jesteśmy w stanie dokonać kodowania ręcznie, na przykład najmniejszą wartość dla edukacji podstawowej, a najwyższą dla edukacji wyższej. Natomiast jeżeli nie znamy tych zależności, możemy wykorzystać LabelEncoder, jednak on ponumeruje klasy w kolejności alfabetycznej, co nie zawsze jest pożądanym rezultatem. Innym podejściem jest One-Hot Encoding[11], który dla każdej z klas tworzy osobną kolumnę z wartością logiczną opisującą, czy dany rekord należy do tej

klasy. Powoduje to wygenerowanie sporej ilości kolumn, jednak mamy wtedy pewność, że nie stworzymy nowych zależności między klasami

2.4 Standaryzacja

Standaryzacja jest procesem, po zakończeniu którego zmienna ma średnią wartość oczekiwaną zero oraz odchylenie standardowe równe jeden, dzięki czemu zyskujemy większą przejrzystość w jej analizie. Bardziej wyraźne są skupienia wokół konkretnych wartości, jednak należy zadbać o to, aby przed standaryzacją pozbyć się wartości odstających, gdyż będą one miały negatywny wpływ na zmienną po standaryzacji

Rozdział 3

Środowisko wykonawcze

W poniższym rozdziale przedstawione zostało środowisko wykonawcze, w jakim przeprowadzono eksperymenty, a także opisano zbiory danych, na których je przeprowadzono

3.1 Specyfikacja sprzetowa, system i środowisko

Sprzęt: Laptop wyposażony w procesor Intel Core i5-1135G7 (2.4Ghz) ze zintegrowaną grafiką oraz 16GB pamięci RAM System operacyjny: Windows 10 Education Menadżer środowisk: Anaconda Navigator Python: 3.9.12 Edytor kodu: Visual Studio Code z dodatkami do edycji plików w formacie Jupyter notebook

3.2 Zbiory danych

Zbiory na których będziemy sprawdzać wpływ przygotowania danych są zbiorami do klasyfikacji. Do eksperymentów wybrano następujące zestawy danych:

Zbiory danych			
Nazwa zbioru	Ilość rekordów	Kolumny numeryczne	Kolumny kategoryczne
League of Legends Stats: S13[2]	485	3	7
Rain in Australia[3]	16443	16	7
Titanic[1]	891	5	5

3.2.1 League of Legends stats

Zbiór zawierający statystyki postaci z gry League of Legends z dwóch wersji obecnego sezonu (13.1 oraz 13.3)

3.2.2 Australian Rain Forecast

Zbiór zawiera codzienne obserwacje dotyczące pogody z różnych lokalizacji na terenie Australii

3.2.3 Titanic Survival

Jest to zestaw informacji na temat pasażerów Titanica oraz tego, czy udało im się przeżyć, na podstawie czego budujemy model, który próbuje przewidzieć na podstawie informacji które mu przekażemy, czy dana osoba przeżyła katastrofę

3.3 Przygotowanie danych do eksperymentów

3.3.1 Część wspólna dla wszystkich zbiorów danych

Do przygotowania danych do eksperymentów służyła funkcja prepare_to_file (Przykład 3.1), która losowo generowała braki w danych, a następnie zapisywała nowopowstały niepełny zbiór danych do pliku.

```
    def prepare_to_file(df,work_columns,filename,count):
    path=r'C:\Users\Darek\Documents\Magisterka
␣␣␣␣\PracaMagPreproccessing\Datasets\Prepared'
    for i in range(count):
        df_copy = df.copy()
        df_random_rows = df.sample(frac=0.1)
        random_indexes = df_random_rows.index.tolist()
        for j in random_indexes:
            chosen_column = random.choice(work_columns)
            df_copy.at[j, chosen_column] = pd.NA
        df_copy.to_csv(path + '\\' + filename +
        '_'+str(i+1)+'.csv', index=False)
```

Przykład 3.1: Funkcja generująca braki w podanym zbiorze danych

3.3.2 Zbiór danych Titanic

Dla zestawu danych Titanic wypełniono brakujące wartości dla wieku oraz portu, w którym pasażer wsiadł na statek (Przykład 3.2)

```
titanic['Age'] = titanic['Age']
.fillna(titanic['Age'].mean())
titanic['Embarked'] = titanic['Embarked']
.fillna('S')
```

Przykład 3.2: Przygotowanie zbioru danych Titanic do eksperymentów

3.3.3 Zbiór danych Lol Stats

Połączono dostępne zbiory danych dla wersji 13.1 oraz 13.3, dodano odpowiednią klasę dla bohatera K'Sante, dokonano odpowiedniego kodowania atrybutu klasyfikującego ,a także sformatowano kolumny zawierające wartości procentowe

```
lol_stats = pd.concat([lol_13_1,lol_13_3])
for i in lol_stats.loc[lol_stats['Name'] == "K'Sante"].index.values:
    lol_stats.at[i,"Class"] = "Tank"
dict = {"God" : '0',
          "S" : '1',
          "A": '2',
          "B" : '3',
          "C": '4',
          "D": '5'}
lol_stats=lol_stats.replace({"Tier": dict})
percent_columns = ['Win %', 'Role %',
   'Pick %', 'Ban %']
for col in percent_columns:
    lol_stats[col] = lol_stats[col].str.rstrip('%')
```

Przykład 3.3: Przygotowanie zbioru danych Lol Stats do eksperymentów

3.3.4 Zbiór danych Australian Rain Forecast

Z całego dostępnego zbioru danych wybrano jedynie dane dotyczące stacji pogodowych z Sydney, Lotniska Sydney, Melbourne, Lotniska Melbourne, Canberry, Newcastle oraz Perth, a także zakodowano odpowiednio wartości binarne

```
aus_weather = aus_weather.dropna()
aus_weather = aus_weather[aus_weather['Location'].isin(
    ['Sydney','SydneyAirport',
    'Canberra','MelbourneAirport',
    'Melbourne','Brisbane','Perth'])]
aus_weather['RainToday'] = aus_weather['RainToday']
.replace('Yes', '1')
aus_weather['RainToday'] = aus_weather['RainToday']
.replace('No', '0')
aus_weather['RainTomorrow'] = aus_weather['RainTomorrow']
.replace('Yes', '1')
aus_weather['RainTomorrow'] = aus_weather['RainTomorrow']
.replace('No', '0')
```

Przykład 3.4: Przygotowanie zbioru danych Australian Rain Forecast do eksperymentów

Rozdział 4

Wykonane Eksperymenty

Dla każdego z wybranych zbiorów danych przeprowadzono przygotowanie danych według ustalonego scenariusza, a także dodatkowego scenariusza w którym głównym celem było indywidualne podejście do zbioru oraz wyciągnięcie możliwie jak najwięcej informacji z dostępnych danych

4.1 Użyte klasyfikatory

4.1.1 xgBoost

xgBoost to zoptymalizowana, zbiorowa biblioteka wzmacniająca gradient, zaprojektowana aby być wysoce efektywną, elastyczną oraz współpracującą z wieloma rodzajami sprzętu. XGBoost korzysta z równoległego wzmacniania drzew, dzięki którym rozwiązuje wiele problemów z dziedziny data science w szybki i dokładny sposób. [21]

4.1.2 Las Losowy

Wykorzystano RandomForestClassifier z biblioteki scikitlearn. Jest to algorytm uczenia maszynowego metodą zespołową, która polega na tworzeniu wielu drzew decyzyjnych podczas uczenia modelu. Przy przypisywaniu obiektu do danej klasy wybierana jest ta, którą wybrała największa ilość drzew. Las losowy eliminuje tendencje drzew decyzyjnych do nadmiernego dopasowywania się do zbioru uczącego.

4.1.3 k-Najbliższych Sąsiadów

Wykorzystano KNeighborsClassifier z biblioteki scikitlearn. Danymi wejściowymi jest k najbliższych sąsiadów w zbiorze danych. Wyjściem jest przynależność do klasy danego obiektu. Obiekt jest klasyfikowany przez głosowanie większościowe spośród k-najbliższych ze zbioru uczącego. Zazwyczaj metryką

używaną do głosowania jest odległość euklidesowa, jednak nie jest to jedyna możliwość.

4.2 Brak Preprocessingu

Pierwszy scenariusz, który w przyszłości będzie punktem odniesienia polegał na usunięciu rekordów, w których występowały jakiekolwiek brakujące wartości

```
def no_preprocessing(df,num):
    df_1 = df.copy()
    df_1 = remove_missing(df_1)
    y = df_1['Survived']
    df_1= drop_columns(df_1,categorical)
    df_1 = df_1.apply(pd.to_numeric)
    X = df_1
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.1, random_state=42)
    score("Titanic",
        num,
        "No Preprocessing",
        X_train,y_train,X_test,y_test)
```

Przykład 4.1: Brak przygotowania danych dla zbioru danych Titanic

4.3 Wypełeninie brakujących wartości średnią

Drugi scenariusz zakładał wypełnienie brakujących wartości średnią za pomocą metod biblioteki pandas fillna() oraz mean()

```
def fill_missing_mean(df, work_columns):
    for col in work_columns:
        df[col] = df[col].fillna(df[col].mean())
    return df
```

Przykład 4.2: Wypełnienie brakujących wartości średnią

4.4 Wypełeninie brakujących wartości minimum

Kolejny scenariusz zakładał wypełnienie brakujących wartości średnią za pomocą metod biblioteki pandas fillna() oraz min()

```
def fill_missing_max(df, work_columns):
    for col in work_columns:
        df[col] = df[col].fillna(df[col].min())
    return df
```

Przykład 4.3: Wypełnienie brakujących wartości minimum

4.5 Wypełeninie brakujących wartości maksimum

Czwarty scenariusz zakładał wypełnienie brakujących wartości średnią za pomocą metod biblioteki pandas fillna() oraz max()

```
def fill_missing_max(df, work_columns):
    for col in work_columns:
        df[col] = df[col].fillna(df[col].max())
    return df
```

Przykład 4.4: Wypełnienie brakujących wartości maksimum

4.6 Wypełeninie brakujących wartości za pomocą regresji liniowej

W tym scenariuszu wykorzystano pozostałe rekordy, aby przy pomocy regresji liniowej wygenerować brakujące wartości

```
def fill_missing_regression(df, numeric):
    for col in numeric:
        df_num = df[numeric]
        test_data = df_num[df_num[col].isnull()]
        df_num = df_num.dropna()
        x_train = df_num.drop(col,axis=1)
        y_train = df_num[col]
        lr = LinearRegression()
        lr.fit(x_train,y_train)
        test_col = []
        for i in numeric:
            if(i != col):
                test_col.append(i)
        x_test = test_data[test_col]
        x_test = fill_missing_mean(x_test,test_col)
        y_pred = lr.predict(x_test)
        test_data[col] = y_pred
        for i in test_data.index.values:
            df.at[i,col] = test_data.loc[i][col]
        return df
```

Przykład 4.5: Wypełnienie brakujących wartości za pomocą regresji liniowej

4.7 Standaryzacja

Dzięki StandardScaler z biblioteki sklearn dokonano standaryzacji kolumn liczbowych

```
def standardize(df, work_columns):
    standard_scaler = preprocessing.StandardScaler()
    for col in work_columns:
        values = df[col].values
        df_scaled = standard_scaler.fit_transform(
            values.reshape(-1, 1))
        df_scaled = pd.DataFrame(df_scaled)
        df[col] = df_scaled
    return df
```

Przykład 4.6: Standaryzacja kolumn liczbowych

4.8 Standaryzacja oraz skalowanie do przedziału (0,1)

Dzięki StandardScaler z biblioteki sklearn dokonano standaryzacji wraz ze skalowaniem do przedziału (0,1) kolumn liczbowych

```
def normalize(df, work_columns):
    min_max_scaler = preprocessing.MinMaxScaler()
    for col in work_columns:
        values = df[col].values
        df_scaled = min_max_scaler.fit_transform(values.reshape(-1, 1))
        df_scaled = pd.DataFrame(df_scaled)
        df[col] = df_scaled
    return df
```

Przykład 4.7: Skalowanie do przedziału (0,1)

4.9 Skalowanie do przedziału (0,1) oraz usunięcie wartości odstających

Poza skalowaniem z poprzedniego scenariusza, wykorzystano algorytm LocalOutlierFactor do usunięcia wartości odstających

```
def remove_outliers_lof(df,work_columns):
    df_temp = df
    df_temp = df_temp.loc[:, work_columns]
    clf = LocalOutlierFactor(n_neighbors=2)
    clf.fit(df_temp)
    y_pred_outliers = clf.fit_predict(df_temp)
    df_temp['outlier'] = y_pred_outliers

    df_temp = df_temp.loc[df_temp['outlier'] == 1]
    df_temp.drop('outlier', axis=1, inplace=True)
```

```
    df_temp = df_temp.reset_index(drop=True)
    df = df[df.index.isin(df_temp.index)]
    return df
```

Przykład 4.8: Usuwanie wartości odstających

4.10 Kodowanie wartości kategorycznych

Wykorzystano LabelEncoder z biblioteki sklearn do zakodowania wartości kategorycznych na liczbowe. W przypadku, kiedy dla danej kolumny brakowało wartości, rekord usuwano

```
def encode_categorical(df, work_columns):
    encoder = LabelEncoder()
    for col in work_columns:
        df[col] = encoder.fit_transform(df[col])
    return df
```

Przykład 4.9: Usuwanie wartości odstających

4.11 Kodowanie wartości kategorycznych oraz wypełnienie brakujących wartości średnią

Po wykonaniu kodowania z poprzedniego scenariusza, wypełniono brakujące wartości wartością średnią

4.12 Scenariusz indywidualnego podejścia do zbioru danych

4.12.1 League of Legends stats

Dla statystyk z gry League of Legends jedyne co mogło być zrobione poza operacjami z poprzednich scenariuszy było usunięcie kolumny z imieniem postaci, jako że nie wnosiła ona konkretnych informacji, a głównie identyfikowała dany rekord

```
def custom_scenario(df,num):
    df_10 = df.copy()
    df_10 = fill_missing_mean(df_10, numeric)
    df_10 = remove_outliers_lof(df_10, numeric)
    df_10 = normalize(df_10, numeric)
    df_10= drop_columns(df_10,['Name'])
    df_10 = encode_categorical(df_10,to_be_encoded)
    y = df_10['Tier']
```

```
    df_10 = df_10.apply(pd.to_numeric)
    X = df_10
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.1, random_state=42)
    score(
        "LoL_Stats",
        num,
        "Custom_preprocessing",
        X_train,y_train,X_test,y_test)
```

Przykład 4.10: Indywidualny scenariusz dla zestawu danych LoL Stats

4.12.2 Australian Rain Forecast

Dla Australian Rain Forecast rozbito datę na dzień miesiąca, miesiąc oraz rok, dzięki czemu w teorii możemy zaobserwować trendy dotyczące na przykład poszczególnych miesięcy na przestrzeni wielu lat

```
def custom_scenario(df,num):
    df_10 = df.copy()
    df_10 = fill_missing_mean(df_10,numeric)
    df_10 = remove_outliers_lof(df_10,numeric)
    df_10 = normalize(df_10,numeric)
# Extract day, month and year from date
    df_10['Year'] = pd.DatetimeIndex(df_10['Date']).year
    df_10['Month'] = pd.DatetimeIndex(df_10['Date']).month
    df_10['Day'] = pd.DatetimeIndex(df_10['Date']).day
    df_10= drop_columns(df_10,['Date'])
    to_be_encoded = [
        'Location',
        'WindGustDir',
        'WindDir9am',
        'WindDir3pm',
        'RainToday',
        'Day',
        'Month',
        'Year']
    df_10 = encode_categorical(df_10,to_be_encoded)
    y = df_10['RainTomorrow']
    df_10 = df_10.apply(pd.to_numeric)
    X = df_10
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.1, random_state=42)
    score(
        "Aus_weather",num,"Custom_preprocessing",
```

```
        X_train, y_train, X_test, y_test)
```

Przykład 4.11: Indywidualny scenariusz dla zestawu danych Aus Weather

4.12.3 Titanic Survival

Dla zestawu danych Titanic wyciągnięto informację o tytule, jakim posługiwał się dany pasażer, dzięki czemu uzyskaliśmy informację o grupie społecznej, do której należeli pasażerowie. Dokonano również zmiany informacji o kabinie, którą zajmował pasażer, znacznie cenniejszą informacją od konkretnej kabiny jest sektor, w którym ona się znajdowała, informacja ta została uzyskana poprzez ograniczenie kodu kabiny jedynie do pierwszego znaku, który określa sektor

```
def custom_scenario(df,num):
    df_10 = df.copy()
    df_10['Title'] = df_10['Name'].str.extract(
        ' ([A-Za-z]+)\.', expand=False)
    df_10['Title'] = df_10['Title'].fillna(
        df_10['Title'].mode().iloc[0])
    df_10['Title'] = df_10['Title'].replace([
        'Lady', 'Countess','Capt', 'Col','Don', 'Dr',\
        'Major', 'Rev', 'Sir', 'Jonkheer', 'Dona'], 'Rare')
    df_10['Title'] = df_10['Title'].replace('Mlle', 'Miss')
    df_10['Title'] = df_10['Title'].replace('Ms', 'Miss')
    df_10['Title'] = df_10['Title'].replace('Mme', 'Mrs')
    df_10 = df_10.drop(['Name','Ticket','PassengerId'],axis=1)
    df_10['Cabin'] = df_10['Cabin'].fillna('000')
    df_10['Cabin'] = df_10['Cabin'].str[:1]
    df_10 = fill_missing_mean(df_10,numeric)
    df_10 = df_10.fillna(df_10.mode().iloc[0])
    to_be_encoded = ["Sex","Embarked","Cabin","Title"]
    df_10 = encode_categorical(df_10,to_be_encoded)
    y = df_10['Survived']
    df_10 = df_10.apply(pd.to_numeric)
    X = df_10
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.1, random_state=42)
    score(
        "Titanic",num,"Custom_preprocessing",
        X_train,y_train,X_test,y_test)
```

Przykład 4.12: Usuwanie wartości odstających

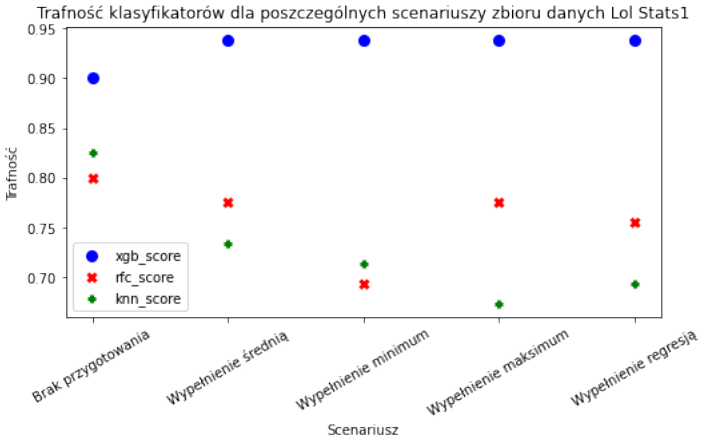
Rozdział 5

Wyniki Eksperymentów

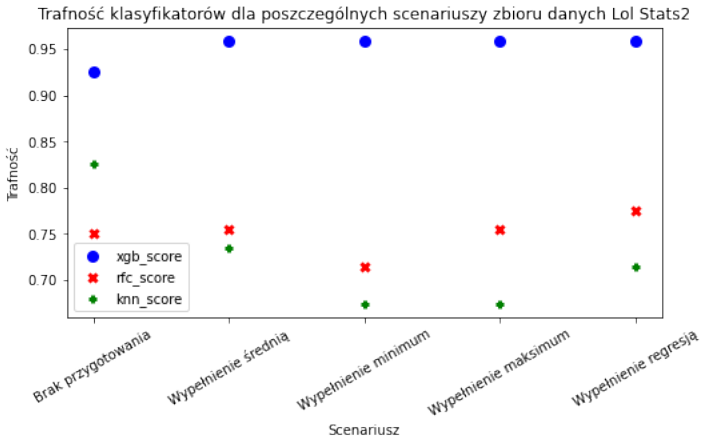
W poniższym rozdziale przedstawiono wyniki eksperymentów dla każdego z klasyfikatorów, z podziałem na zbliżone do siebie scenariusze, jak i również średnie wyniki dla każdego ze zbiorów danych.

5.1 League of Legends stats

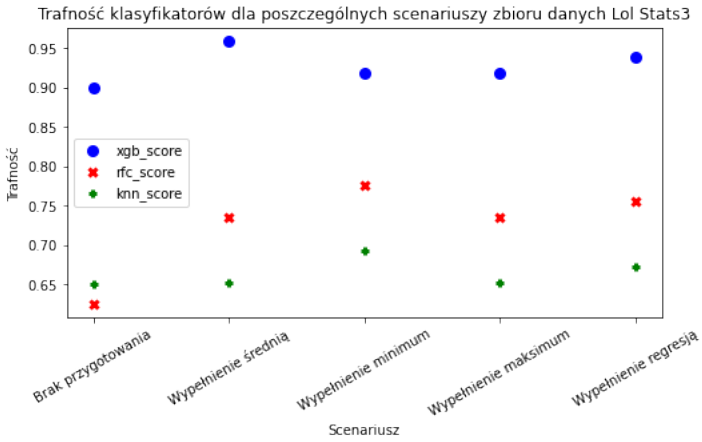
5.1.1 Wypełnienie brakujących wartości



Rysunek 5.1: Trafność klasyfikatorów po wypełnieniu brakujących wartości dla zestawu danych Lol Stats 1



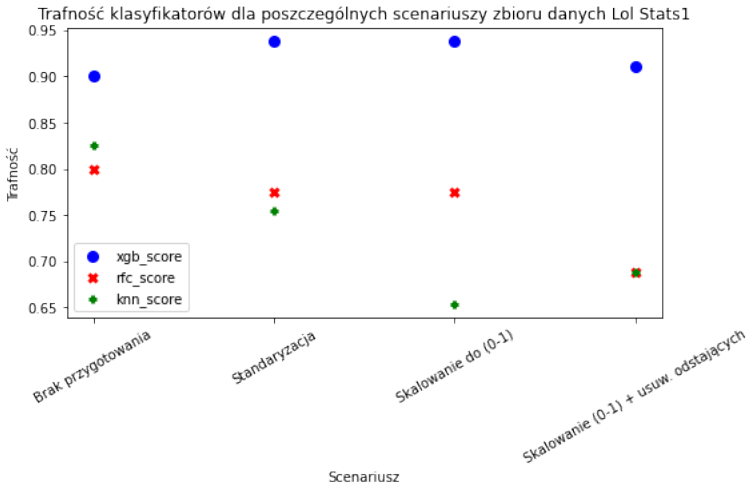
Rysunek 5.2: Trafność klasyfikatorów po wypełnieniu brakujących wartości dla zestawu danych Lol Stats 2



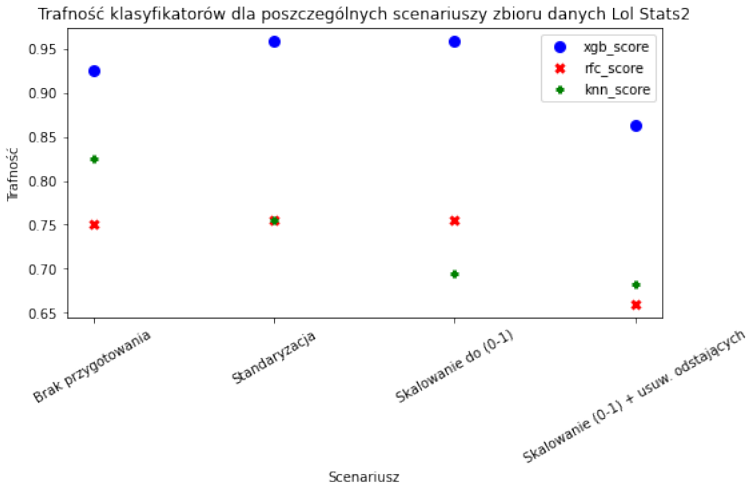
Rysunek 5.3: Trafność klasyfikatorów po wypełnieniu brakujących wartości dla zestawu danych Lol Stats 3

Jak widać na wykresach (Rysunek 5.1, Rysunek 5.2, Rysunek 5.3), dla klasyfikatora xgBoost wypełnienie jakąkolwiek wartością daje widoczną poprawę w trafności, jednak najlepsze wyniki uzyskujemy przy wartości średniej. Dla RandomForestClassifier uzyskujemy różne rezultaty w zależności od wygenerowanych dziur w zbiorze danych, w pierwszym przypadku jakiekolwiek wypełnianie pogarsza klasyfikację, w drugim regresja jest najlepszym sposobem, a w trzecim minimum. Dla algorytmu k-nearest neighbors w każdym z przypadków nie uzyskujemy poprawy albo wręcz pogarszamy wyniki przez wypełnienie brakujących wartości.

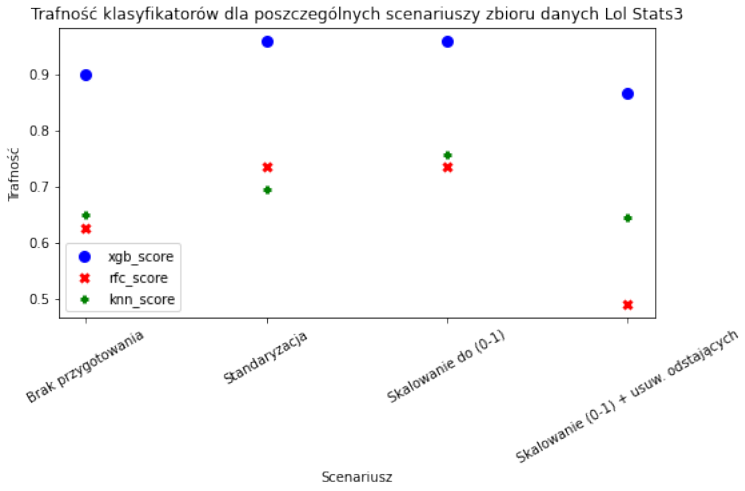
5.1.2 Standaryzacja



Rysunek 5.4: Trafność klasyfikatorów po standaryzacji dla zestawu danych Lol Stats 1



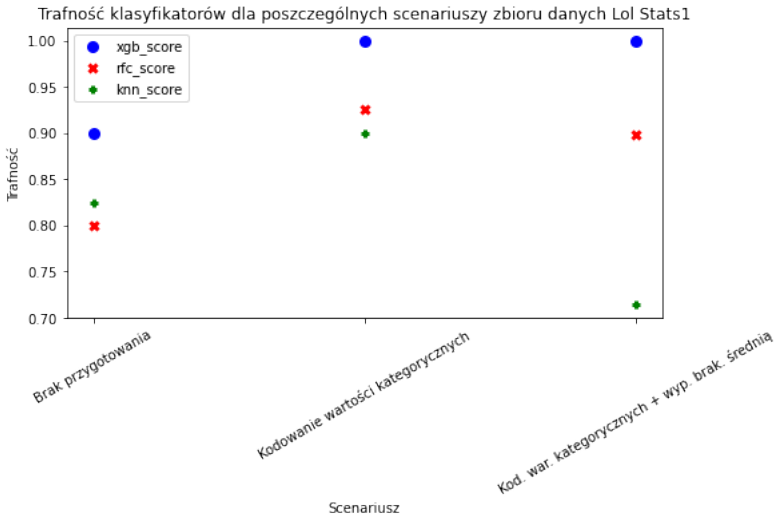
Rysunek 5.5: Trafność klasyfikatorów po standaryzacji dla zestawu danych Lol Stats 2



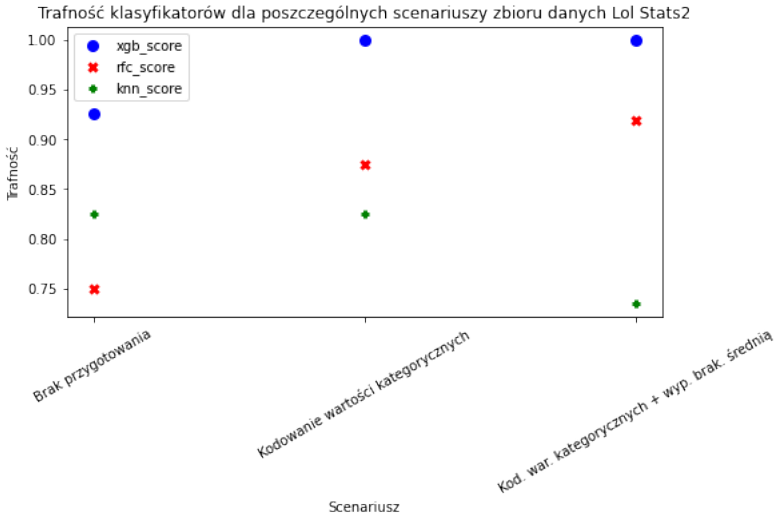
Rysunek 5.6: Trafność klasyfikatorów po standaryzacji dla zestawu danych Lol Stats 3

Na wykresach (Rysunek Klasyfikator xgBoost osiąga niemalże perfekcję po standaryzacji lub standaryzacji ze skalowaniem do przedziału (0,1). Dla RandomForestClassifier jedynie w trzecim przypadku uzyskujemy poprawę przy standaryzacji oraz skalowaniu do przedziału (0,1) Dla algorytmu k-nearest neighbors podobnie jak dla RandomForestClassifier jedynie trzeci przypadek wykazuje poprawę trafności. Należy zwrócić uwagę na fakt, że dla każdego z przypadków usunięcie wartości odstających znacznie pogarsza trafność klasyfikacji, nawet względem braku przygotowania danych

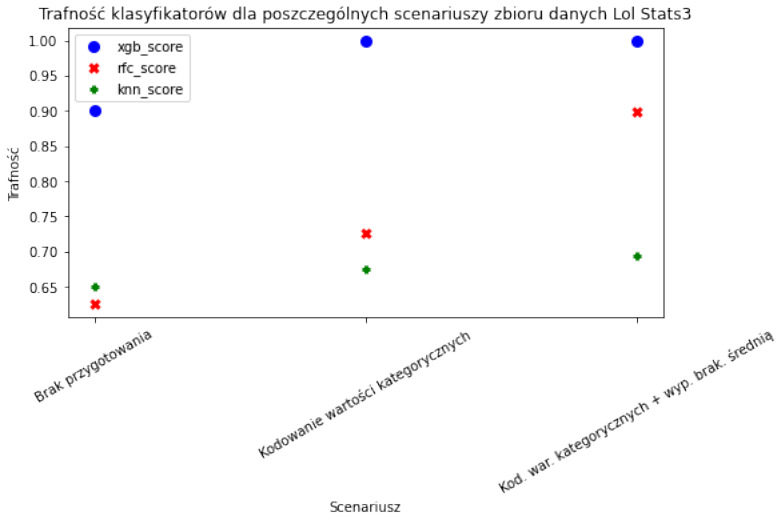
5.1.3 Kodowanie



Rysunek 5.7: Trafność klasyfikatorów po kodowaniu dla zestawu danych Lol Stats 1



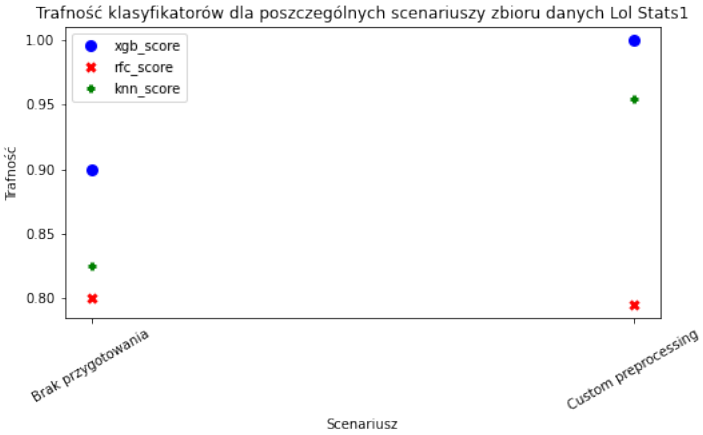
Rysunek 5.8: Trafność klasyfikatorów po kodowaniu dla zestawu danych Lol Stats 2



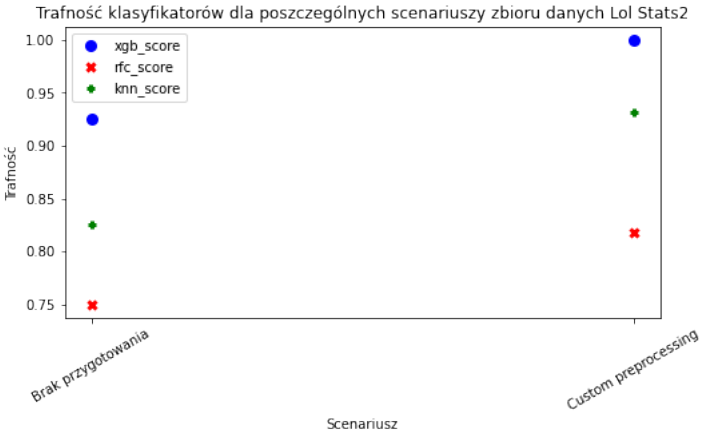
Rysunek 5.9: Trafność klasyfikatorów po kodowaniu dla zestawu danych Lol Stats 3

XgBoost po kodowaniu wartości kategorycznych osiąga perfekcję i utrzymuje się na tym poziomie, RandomForestClassifier dla samego kodowania wykazuje poprawę we wszystkich przypadkach, natomiast dla pierwszego przypadku wypełnienie brakujacych wartości średnią powoduje pogorszenie wyników względem samego kodowania. Dla algorytmu k-nearest neighbors poprawa uzyskana dzięki kodowaniu jest nieznaczna, jednak bardzo często w połączeniu z wypełenieniem wartości brakujących średnią uzyskujemy wynik gorszy od braku przygotowania.

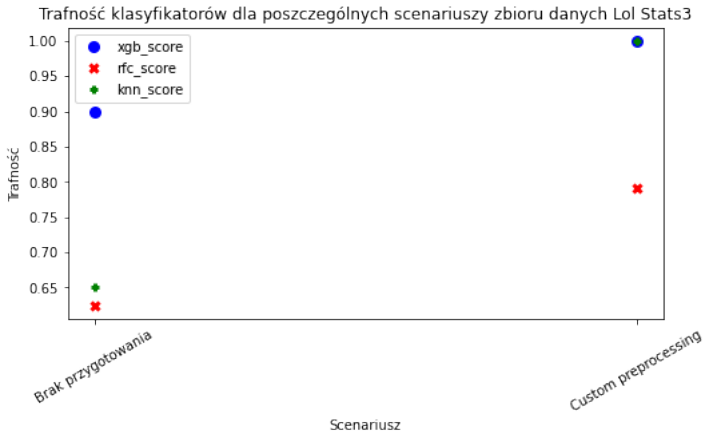
5.1.4 Indywidualne podejście



Rysunek 5.10: Trafność klasyfikatorów po indywidualnym podejściu dla zestawu danych Lol Stats 1



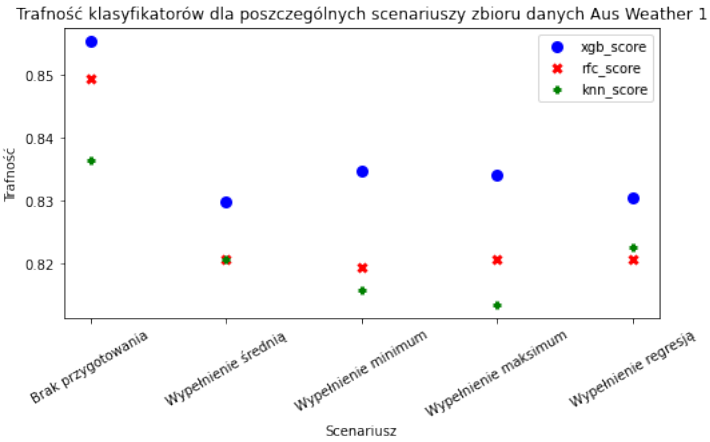
Rysunek 5.11: Trafność klasyfikatorów po indywidualnym podejściu dla zestawu danych Lol Stats 2



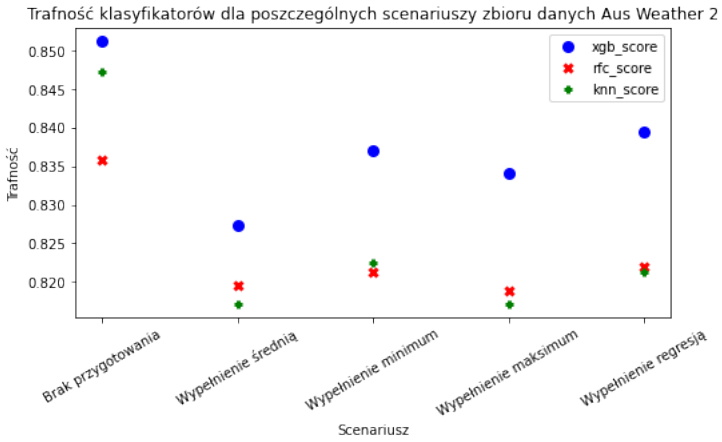
Rysunek 5.12: Trafność klasyfikatorów po indywidualnym podejściu dla zestawu danych Lol Stats 3

5.2 Australian Rain Forecast

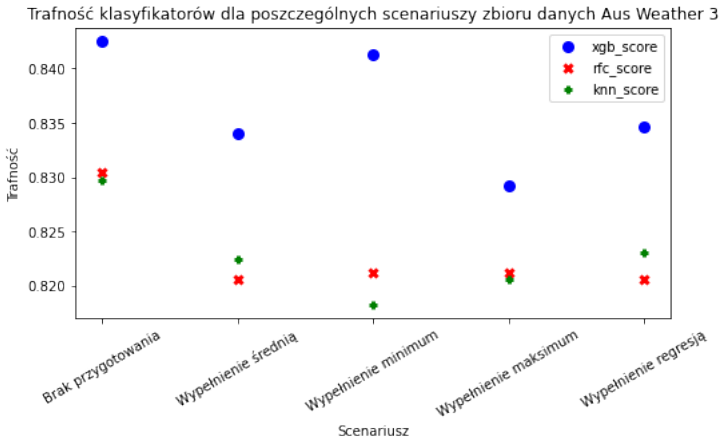
5.2.1 Wypełnienie brakujących wartości



Rysunek 5.13: Trafność klasyfikatorów po przygotowaniu danych według danego scenariusza

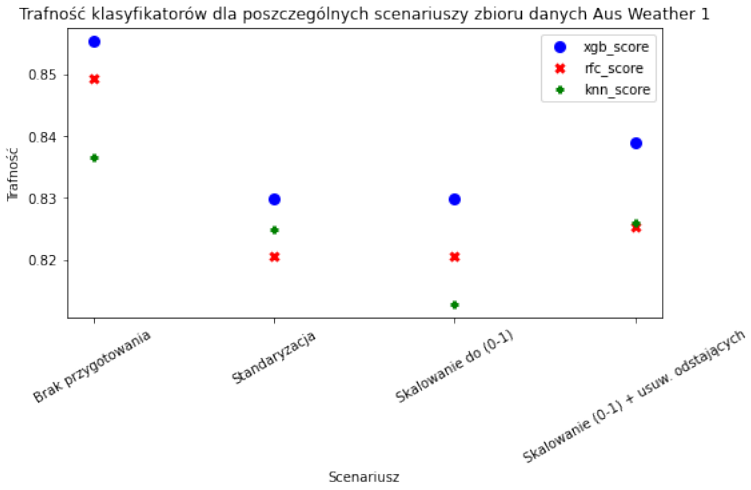


Rysunek 5.14: Trafność klasyfikatorów po przygotowaniu danych według danego scenariusza

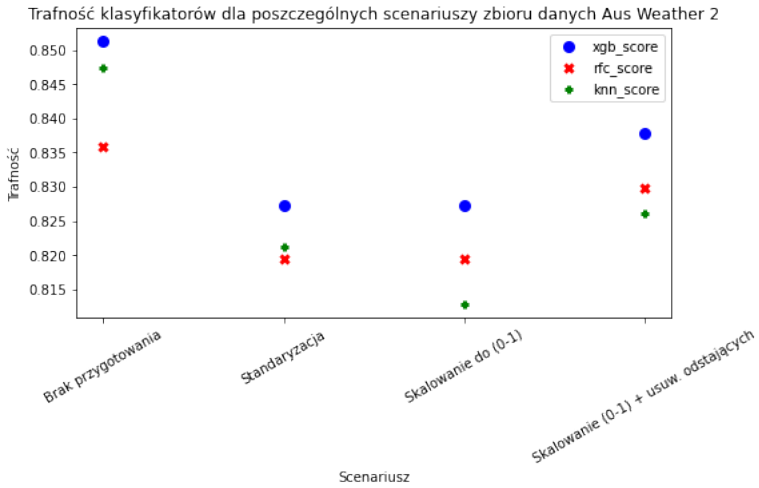


Rysunek 5.15: Trafność klasyfikatorów po przygotowaniu danych według danego scenariusza

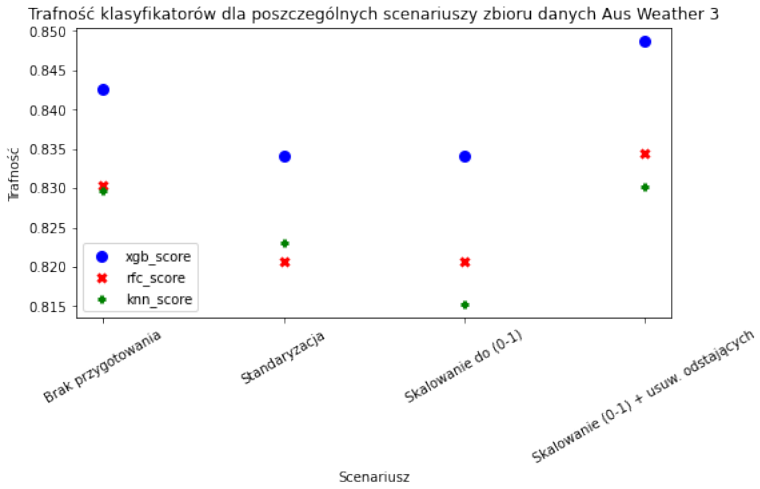
5.2.2 Standaryzacja



Rysunek 5.16: Trafność klasyfikatorów po przygotowaniu danych według danego scenariusza

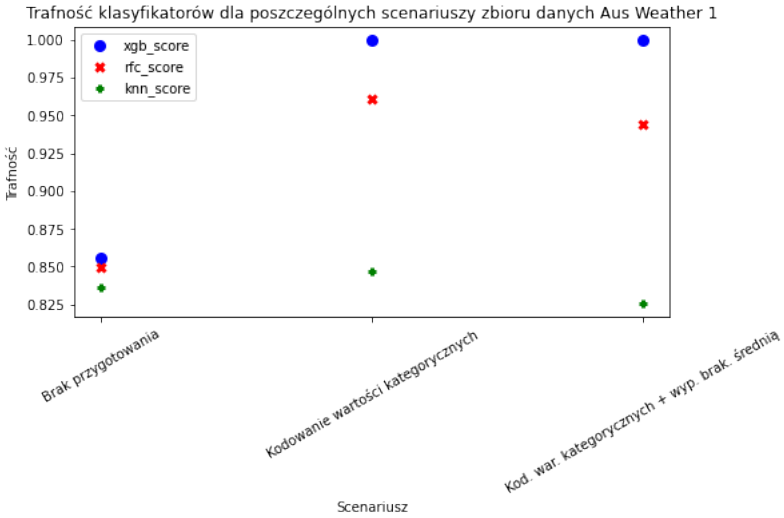


Rysunek 5.17: Trafność klasyfikatorów po przygotowaniu danych według danego scenariusza

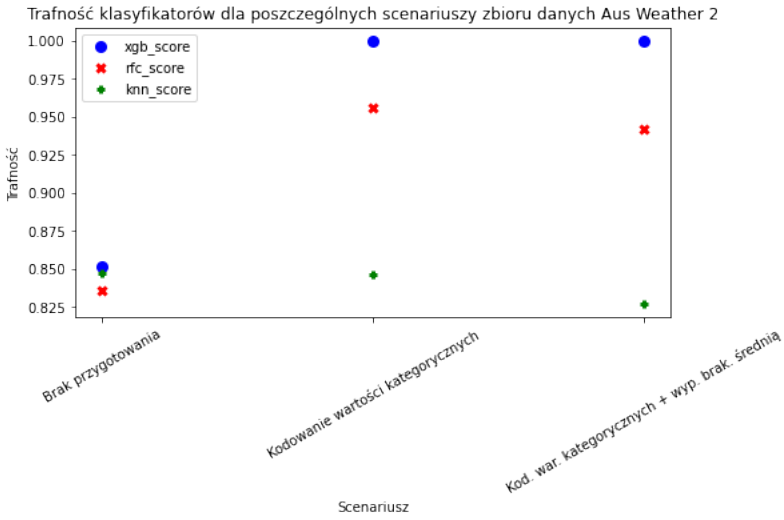


Rysunek 5.18: Trafność klasyfikatorów po przygotowaniu danych według danego scenariusza

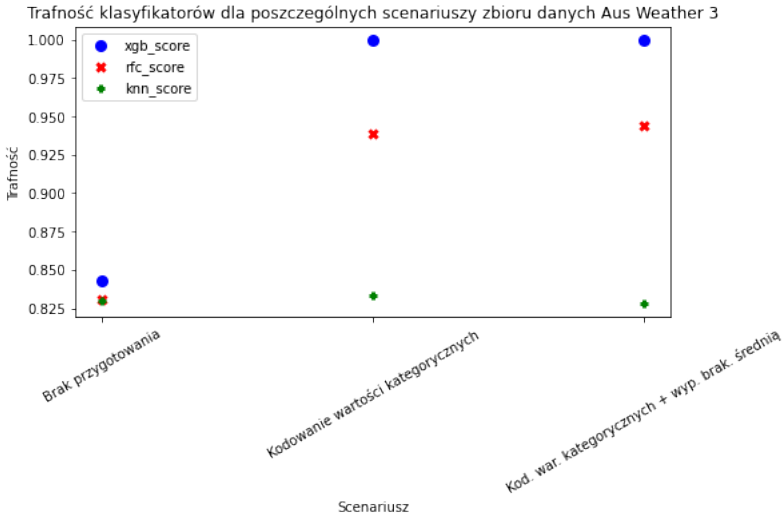
5.2.3 Kodowanie



Rysunek 5.19: Trafność klasyfikatorów po przygotowaniu danych według danego scenariusza

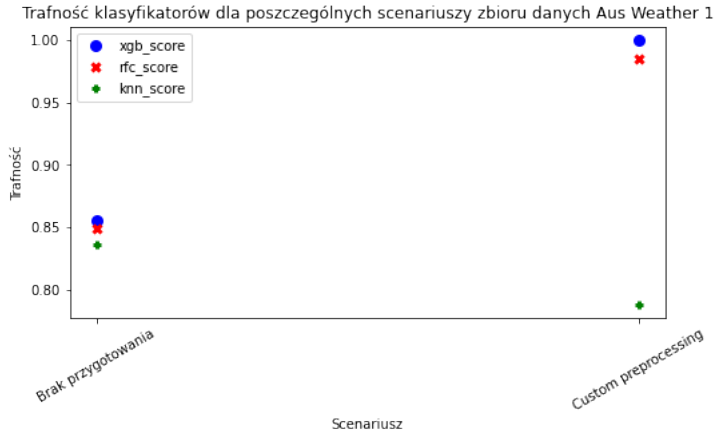


Rysunek 5.20: Trafność klasyfikatorów po przygotowaniu danych według danego scenariusza



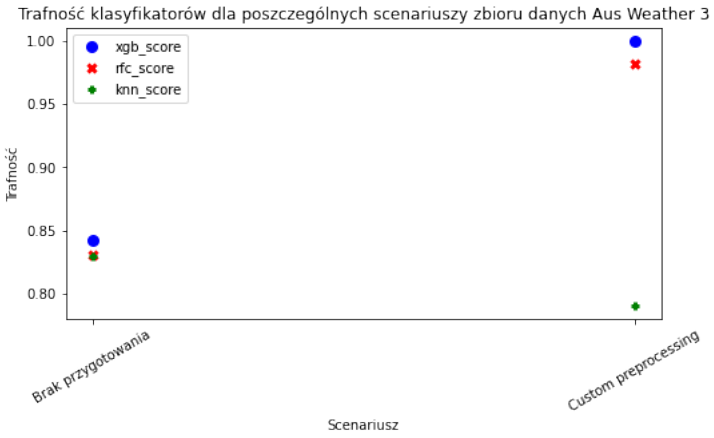
Rysunek 5.21: Trafność klasyfikatorów po przygotowaniu danych według danego scenariusza

5.2.4 Indywidualne podejście



Rysunek 5.22: Trafność klasyfikatorów po przygotowaniu danych według danego scenariusza

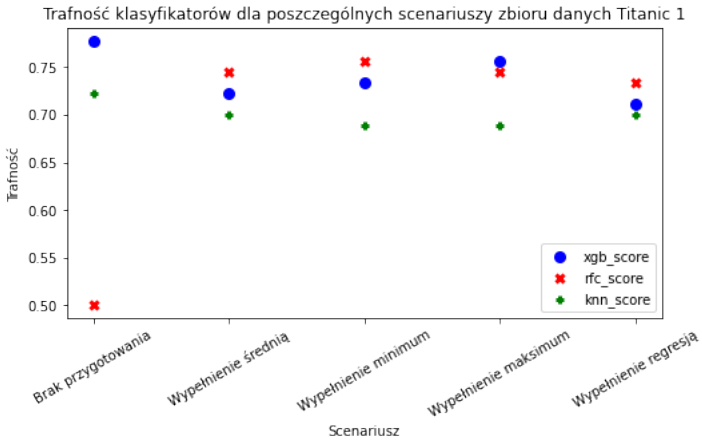
Rysunek 5.23: Trafność klasyfikatorów po przygotowaniu danych według danego scenariusza



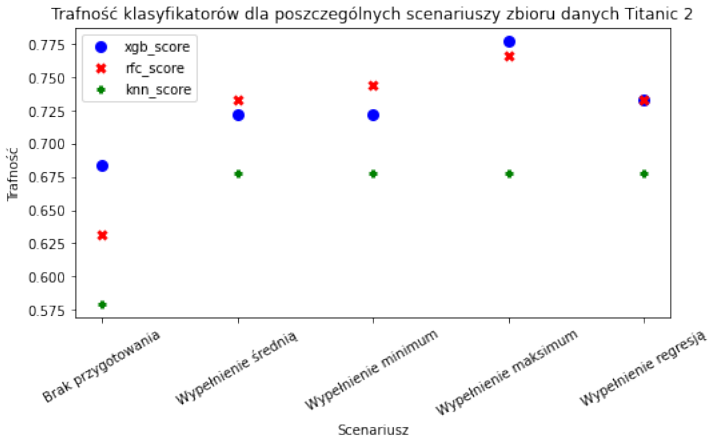
Rysunek 5.24: Trafność klasyfikatorów po przygotowaniu danych według danego scenariusza

5.3 Titanic Survival

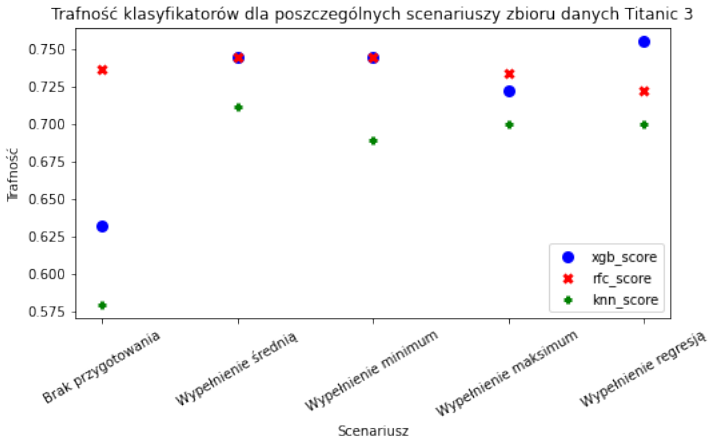
5.3.1 Wypełnienie brakujących wartości



Rysunek 5.25: Trafność klasyfikatorów po przygotowaniu danych według danego scenariusza

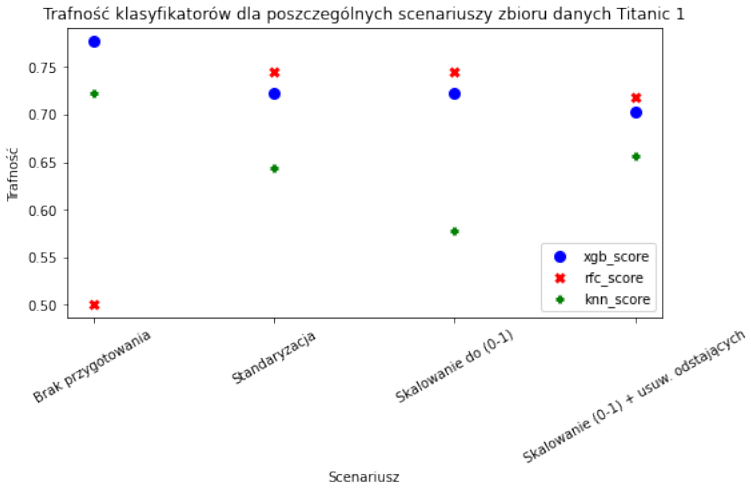


Rysunek 5.26: Trafność klasyfikatorów po przygotowaniu danych według danego scenariusza

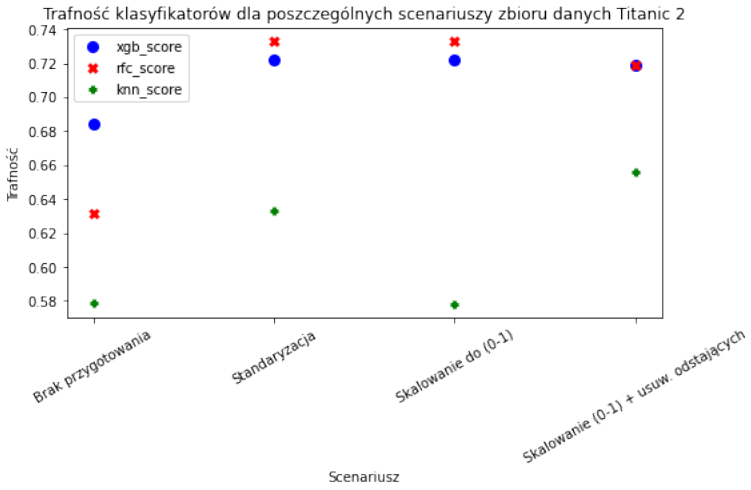


Rysunek 5.27: Trafność klasyfikatorów po przygotowaniu danych według danego scenariusza

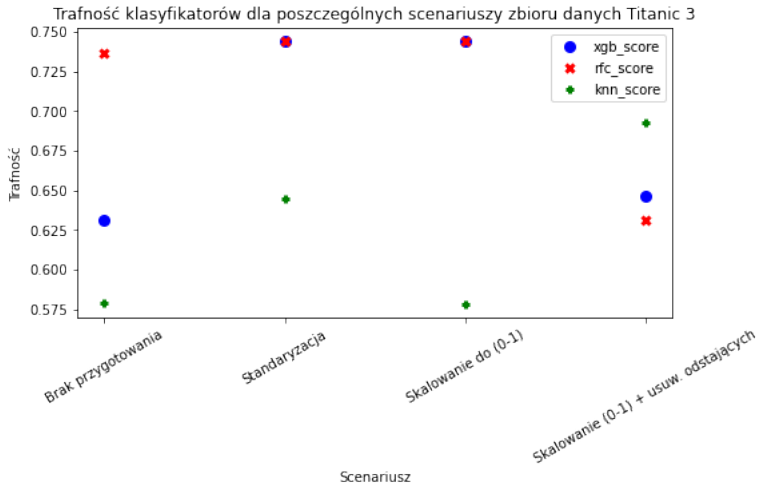
5.3.2 Standaryzacja



Rysunek 5.28: Trafność klasyfikatorów po przygotowaniu danych według danego scenariusza

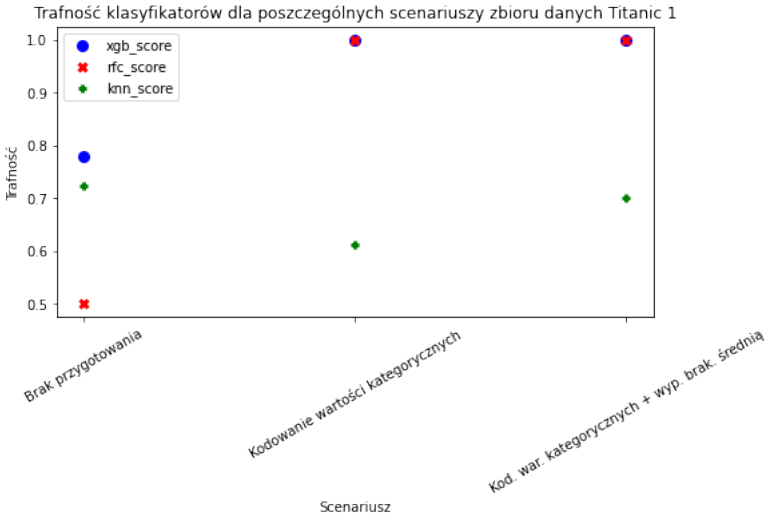


Rysunek 5.29: Trafność klasyfikatorów po przygotowaniu danych według danego scenariusza

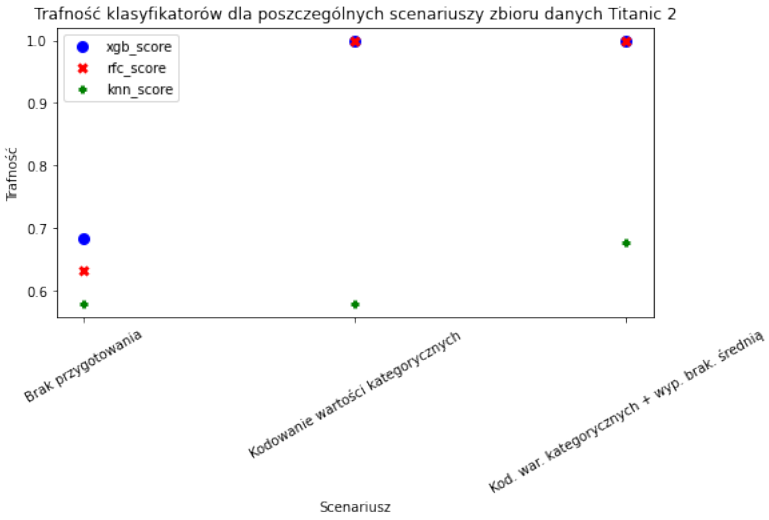


Rysunek 5.30: Trafność klasyfikatorów po przygotowaniu danych według danego scenariusza

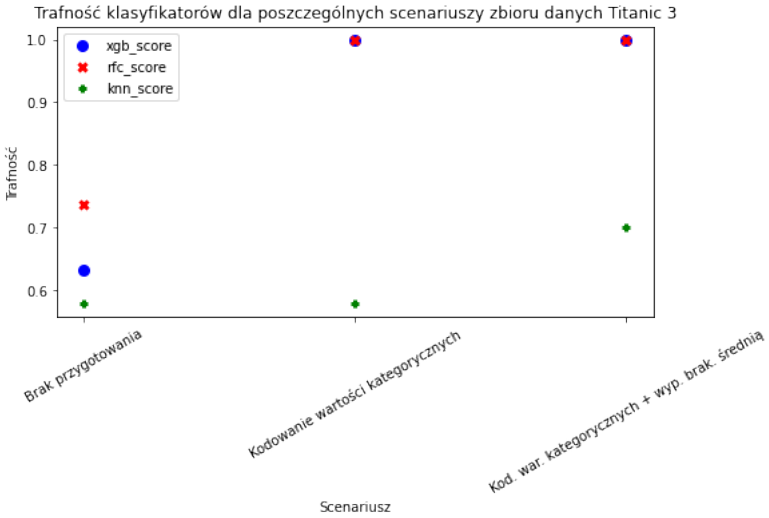
5.3.3 Kodowanie



Rysunek 5.31: Trafność klasyfikatorów po przygotowaniu danych według danego scenariusza

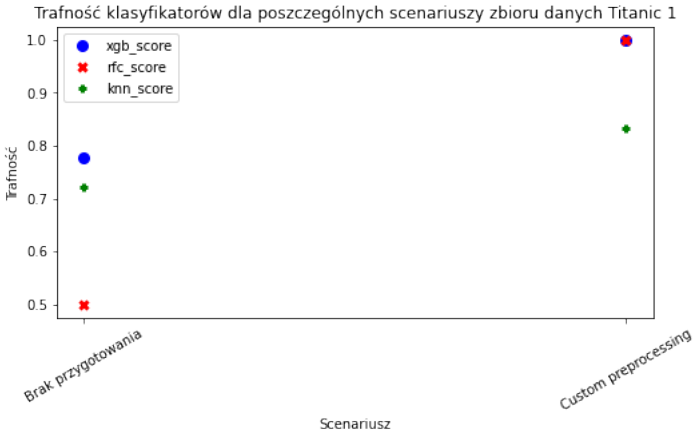


Rysunek 5.32: Trafność klasyfikatorów po przygotowaniu danych według danego scenariusza

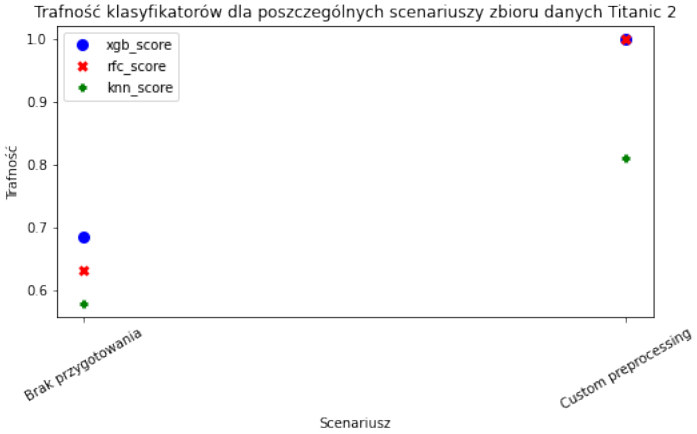


Rysunek 5.33: Trafność klasyfikatorów po przygotowaniu danych według danego scenariusza

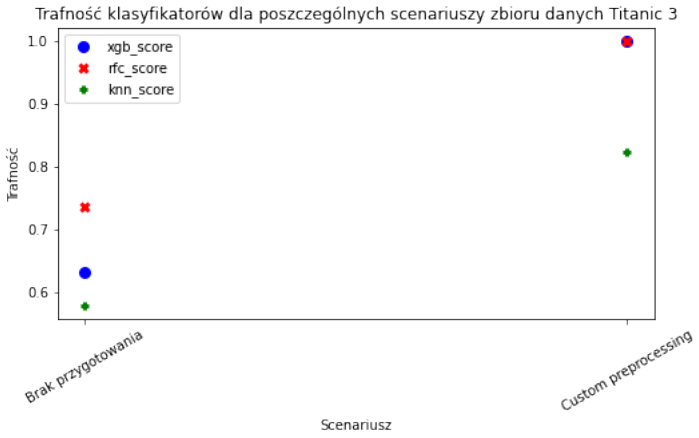
5.3.4 Indywidualne podejście



Rysunek 5.34: Trafność klasyfikatorów po przygotowaniu danych według danego scenariusza



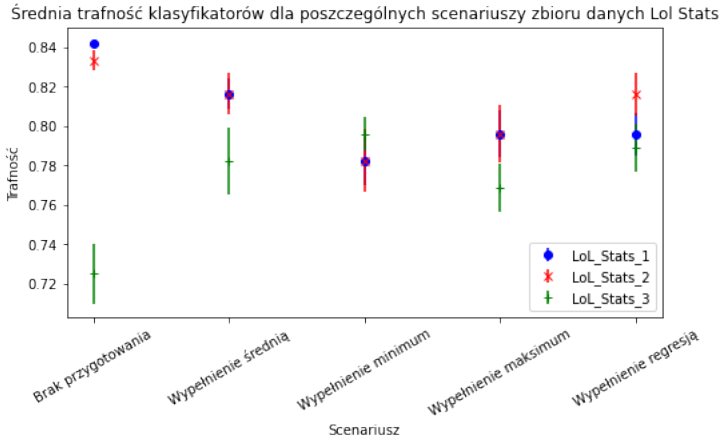
Rysunek 5.35: Trafność klasyfikatorów po przygotowaniu danych według danego scenariusza



Rysunek 5.36: Trafność klasyfikatorów po przygotowaniu danych według danego scenariusza

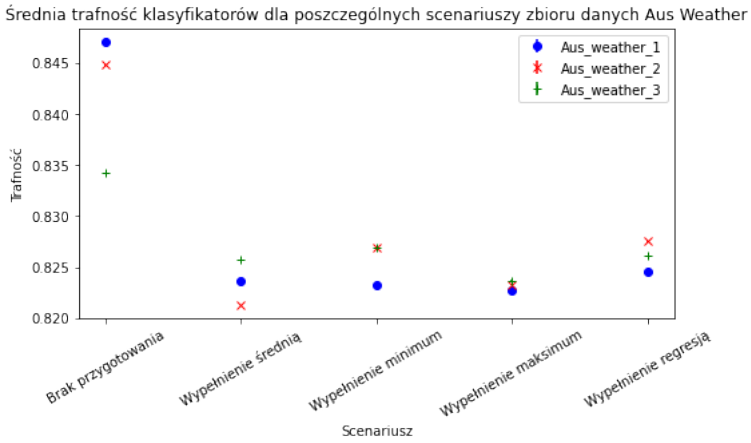
5.4 Średnie wyniki

5.4.1 Wypełnienie brakujących wartości



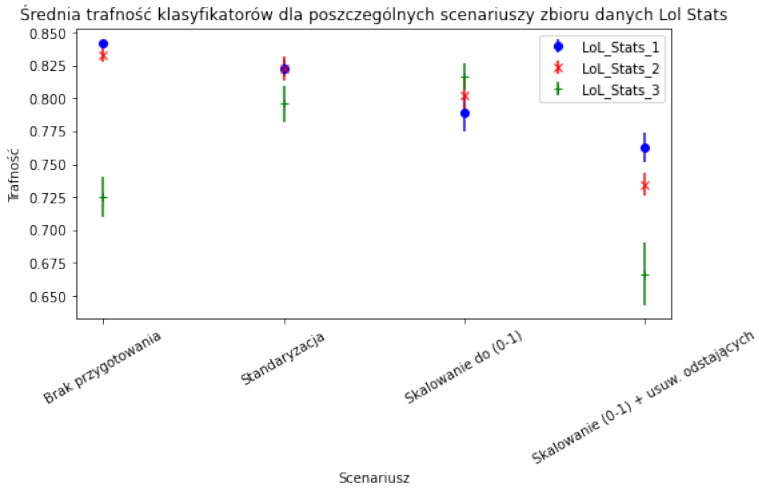
Rysunek 5.37: Średnia trafność klasyfikatorów dla zbioru danych Lol Stats dla grupy scenariuszy wypełniania brakujących wartości

Rysunek 5.38: Średnia trafność klasyfikatorów dla zbioru danych Titanic dla grupy scenariuszy wypełniania brakujących wartości

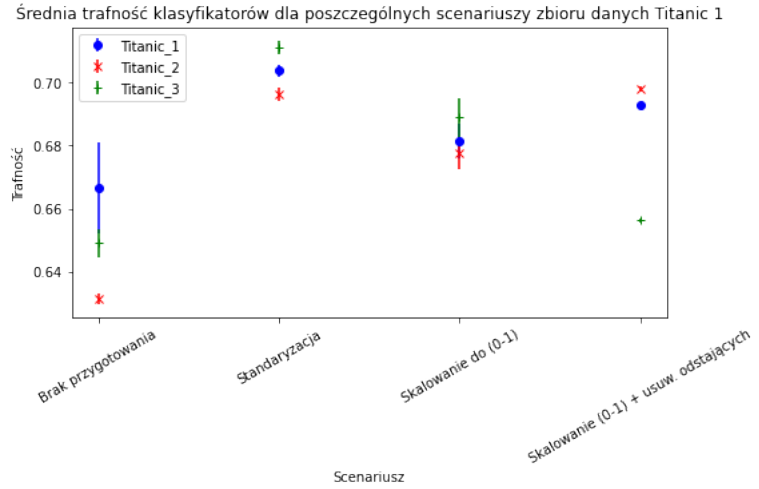


Rysunek 5.39: Średnia trafność klasyfikatorów dla zbioru danych Australian Weather dla grupy scenariuszy wypełniania brakujących wartości

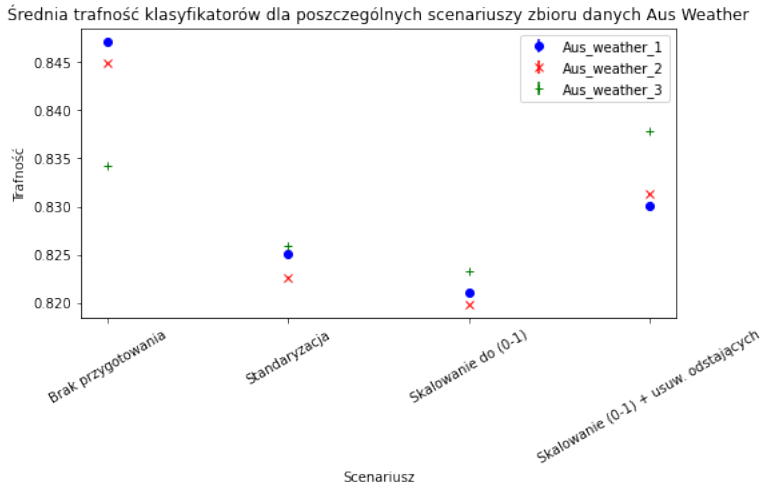
5.4.2 Standaryzacja



Rysunek 5.40: Średnia trafność klasyfikatorów dla zbioru danych Lol Stats dla grupy scenariuszy wypełniania brakujących wartości

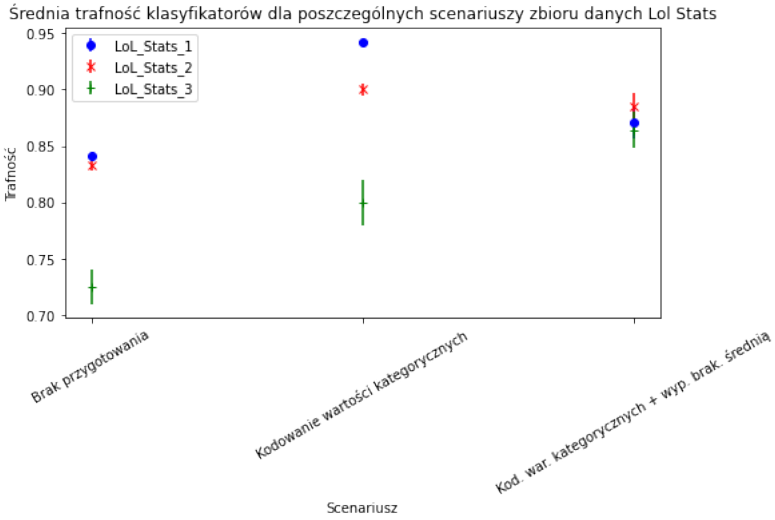


Rysunek 5.41: Średnia trafność klasyfikatorów dla zbioru danych Titanic dla grupy scenariuszy wypełniania brakujących wartości

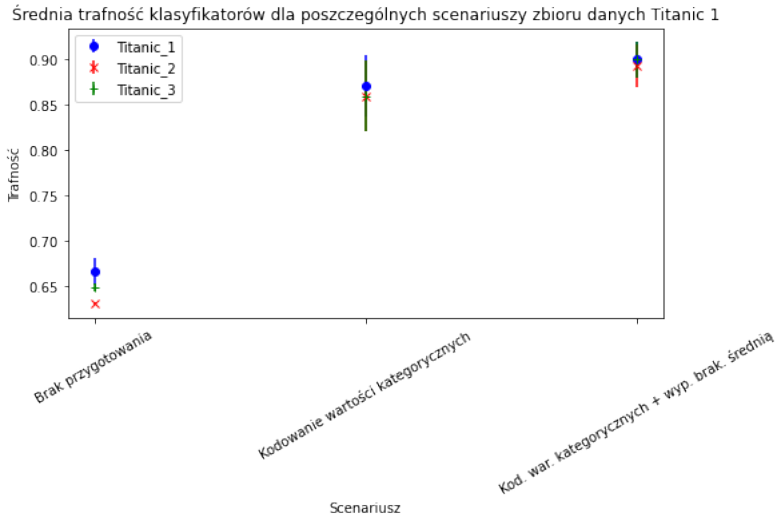


Rysunek 5.42: Średnia trafność klasyfikatorów dla zbioru danych Australian Weather dla grupy scenariuszy wypełniania brakujących wartości

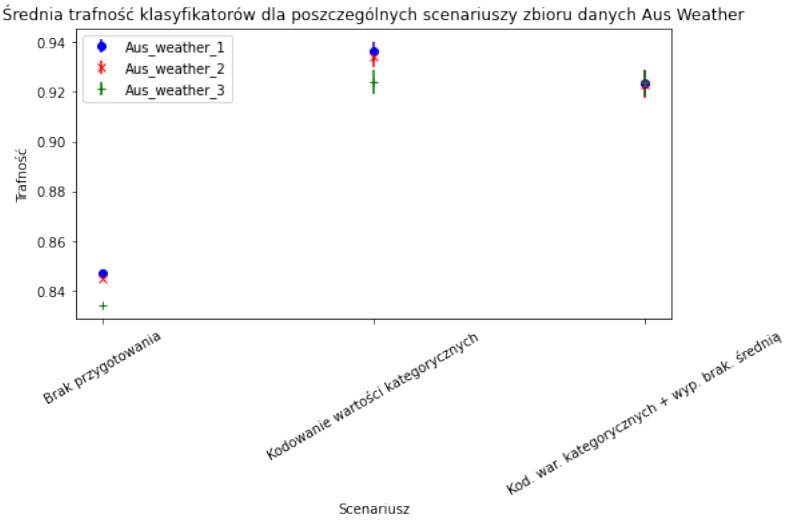
5.4.3 Kodowanie



Rysunek 5.43: Średnia trafność klasyfikatorów dla zbioru danych Lol Stats dla grupy scenariuszy wypełniania brakujących wartości

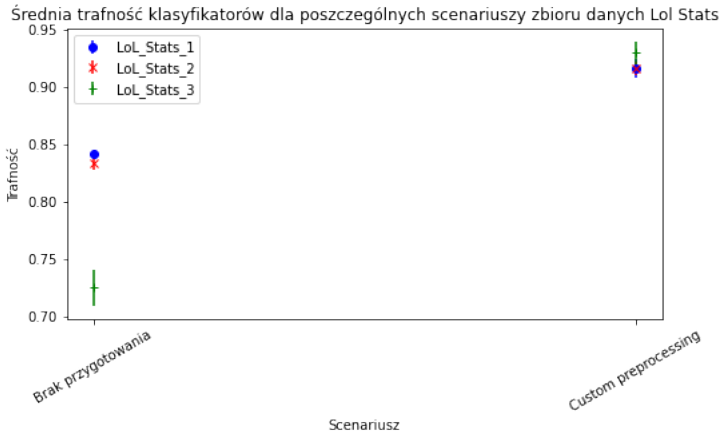


Rysunek 5.44: Średnia trafność klasyfikatorów dla zbioru danych Titanic dla grupy scenariuszy wypełniania brakujących wartości

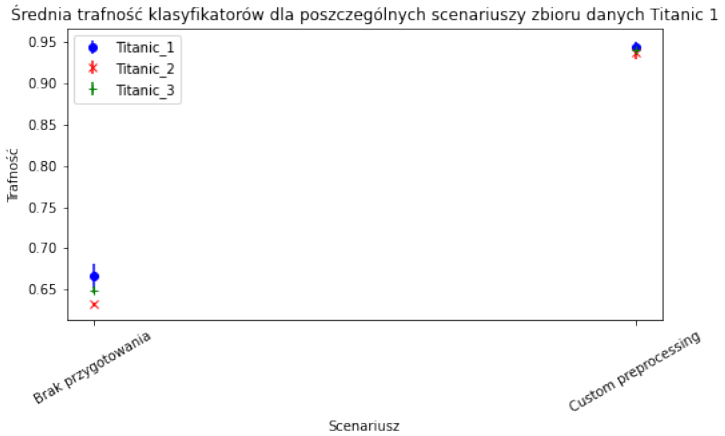


Rysunek 5.45: Średnia trafność klasyfikatorów dla zbioru danych Australian Weather dla grupy scenariuszy wypełniania brakujących wartości

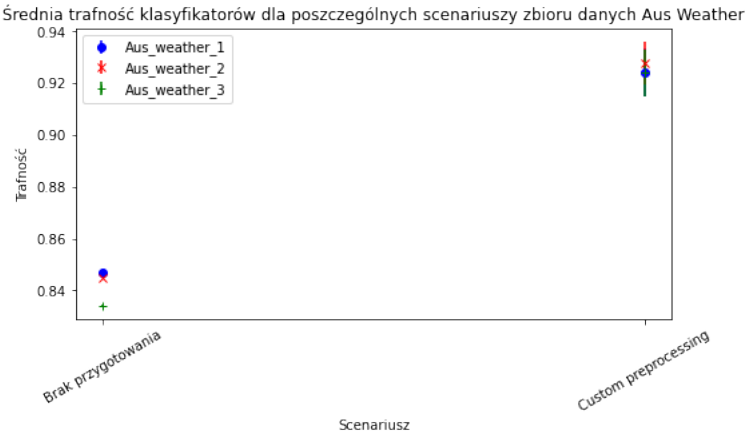
5.4.4 Indywidualne podejście



Rysunek 5.46: Średnia trafność klasyfikatorów dla zbioru danych Lol Stats dla grupy scenariuszy wypełniania brakujących wartości



Rysunek 5.47: Średnia trafność klasyfikatorów dla zbioru danych Titanic dla grupy scenariuszy wypełniania brakujących wartości



Rysunek 5.48: Średnia trafność klasyfikatorów dla zbioru danych Australian Weather dla grupy scenariuszy wypełniania brakujących wartości

Rozdział 6

Podsumowanie

W poniższym rozdziale przedstawiono, jak wyniki eksperymentów mają się do założeń postawionych przed ich przeprowadzeniem, oraz wypisano wnioski powstałe na podstawie wyników eksperymentów.

6.1 Wyniki eksperymentów względem hipotezy

6.2 Podejście dające najlepsze rezultaty

6.3 Konkluzja

Z przeprowadzonych eksperymentów wynika, że odpowiednie przygotowanie danych zwiększa trafność klasyfikatorów. Należy jednak zwrócić uwagę, że nie wszystkie scenariusze skutkowały jednoznaczną poprawą rezultatów. Na szczególne wyróżenienie zasługuje kodowanie wartości kategorycznych na liczbowe, gdyż w każdym przypadku zastosowanie tej metody znacznie zwiększyło trafność klasyfikacji

Bibliografia

[1] Zbiór danych Titanic dostępny do pobrania ze strony kaggle.com

[2] Zbiór danych Lol Stats dostępny do pobrania ze strony kaggle.com

[3] Zbiór danych Australian Rain dostępny do pobrania ze strony kaggle.com

[4] "Transforming Unstructured Data into Useful Information", *Big Data, Mining, and Analytics*, Auerbach Publications, pp. 227–246, 2014-03-12, doi:10.1201/b16666-14, ISBN 978-0-429-09529-0

[5] MOIS, George; FOLEA, Silviu; SANISLAV, Teodora. Analysis of three IoT-based wireless sensors for environmental monitoring. *IEEE Transactions on Instrumentation and Measurement*, 2017, 66.8: 2056-2064.

[6] Margo, Robert A. (2000). *Wages and labor markets in the United States, 1820-1860.* University of Chicago Press. ISBN 0-226-50507-3. OCLC 41285104

[7] Marshall, G. (2005). The purpose, design and administration of a questionnaire for data collection. Radiography, 11(2), 131-136.

[8] Fabijan, A., Olsson, H. H., Bosch, J. (2015). Customer feedback and data collection techniques in software R&D: a literature review. In Software Business: 6th International Conference, ICSOB 2015, Braga, Portugal, June 10-12, 2015, Proceedings 6 (pp. 139-153). Springer International Publishing.

[9] Hasan, M. K., Alam, M. A., Roy, S., Dutta, A., Jawad, M. T., Das, S. (2021). Missing value imputation affects the performance of machine learning: A review and analysis of the literature (2010–2021). Informatics in Medicine Unlocked, 27, 100799.

[10] E. Kreyszig (1979). Advanced Engineering Mathematics (4th ed.). Wiley. p. 880, eq. 5. ISBN 0-471-02140-7.

[11] Okada, S., Ohzeki, M., Taguchi, S. (2019). Efficient partition of integer optimization problems with one-hot encoding. Scientific reports, 9(1), 13036.

[12] Wang, H., Bah, M. J., Hammad, M. (2019). Progress in outlier detection techniques: A survey. Ieee Access, 7, 107964-108000.

[13] Alghushairy, O., Alsini, R., Soule, T., Ma, X. (2020). A review of local outlier factor algorithms for outlier detection in big data streams. Big Data and Cognitive Computing, 5(1), 1.

[14] Potdar, K., Pardawala, T. S., Pai, C. D. (2017). A comparative study of categorical variable encoding techniques for neural network classifiers. International journal of computer applications, 175(4), 7-9.

[15] Narudin, F. A., Feizollah, A., Anuar, N. B., Gani, A. (2016). Evaluation of machine learning classifiers for mobile malware detection. Soft Computing, 20, 343-357.

[16] Khanal, S. S., Prasad, P. W. C., Alsadoon, A., Maag, A. (2020). A systematic review: machine learning based recommendation systems for e-learning. Education and Information Technologies, 25, 2635-2664.

[17] Ezugwu, A. E., Ikotun, A. M., Oyelade, O. O., Abualigah, L., Agushaka, J. O., Eke, C. I., Akinyelu, A. A. (2022). A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. Engineering Applications of Artificial Intelligence, 110, 104743.

[18] Sousa, M. J., Pesqueira, A. M., Lemos, C., Sousa, M., & Rocha, Á. (2019). Decision-making based on big data analytics for people management in healthcare organizations. Journal of medical systems, 43, 1-10.

[19] Schildkamp, K., Lai, M. K., & Earl, L. (Eds.). (2012). Data-based decision making in education: Challenges and opportunities.

[20] Fávero, L. P., & Belfiore, P. (2019). Data science for business and decision making. Academic Press.

[21] Definicja dostępna w dokumentacji biblioteki XGBoost

[22] Cover, Thomas M.; Hart, Peter E. (1967). "Nearest neighbor pattern classification"(PDF). IEEE Transactions on Information Theory. 13 (1): 21–27. CiteSeerX 10.1.1.68.2616. doi:10.1109/TIT.1967.1053964.

[23] Ho, Tin Kam (1995). Random Decision Forests (PDF). Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16 August 1995. pp. 278–282.

[24] Artykuł na temat jakości danych na stronie firmy IBM