Politechnika Wrocławska
Wydział Zarządzania

Techniki Eksploracji Danych

Projekt

Model Drzewa Decyzyjnego

Krzysztof Janicki

Spis treści

1. Opis Danych

1.1 Opis Kolumn

Ten zestaw danych pochodzi pierwotnie z Narodowego Instytutu Cukrzycy, Chorób Trawiennych i Nerek. Celem zestawu danych jest diagnostyczne przewidywanie, czy pacjent ma cukrzycę, na podstawie określonych pomiarów diagnostycznych zawartych w zestawie danych. Na wybór tych przypadków z większej bazy danych nałożono kilka ograniczeń. W szczególności wszyscy pacjenci tutaj to kobiety w wieku co najmniej 21 lat.

```
[3]: df
```

[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0

Informacje o kolumnach zestawu danych:

Pregnancies: Liczba ciąż.

Glucose: Poziom glukozy we krwi.

BloodPressure: Pomiar ciśnienia krwi.

SkinThickness: Grubość skóry.

Insulin: Poziom insuliny we krwi.

BMI: Wskaźnik masy ciała.

DiabetesPedigreeFunction: Wskaźnik dziedziczenia cukrzycy.

Age: Wiek.

Outcome: Wynik końcowy, gdzie 1 oznacza tak chory na cukrzyce, a 0 oznacza nie pacjent jest zdrowy.

1.2 Podstawowe Statystyki Danych

```
[6]: df.describe()
```

[6]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

Widzimy że, mamy brakujące dane w naszym datasecie m.in. w kolumnach Insulin, BMI, SkinThickness, BloodPressure i Glucose. Stworzyliśmy funkcje, która radzi sobie z takimi

wartościami w taki sposób że oblicza medianę danej kolumny i wstawie w miejsce zera takie wartości.

1.2.1 BRAKUJĄCE WARTOŚCI

```
[10]: df.isna().sum()
```

```
[10]: Pregnancies                 0
      Glucose                     0
      BloodPressure               0
      SkinThickness               0
      Insulin                     0
      BMI                         0
      DiabetesPedigreeFunction    0
      Age                         0
      Outcome                     0
      dtype: int64
```

```
[11]: df[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
          'BMI', 'DiabetesPedigreeFunction', 'Age']] = df[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
          'BMI', 'DiabetesPedigreeFunction', 'Age']].replace(0, np.NaN)
```

```
[12]: df.isna().sum()
```

```
[12]: Pregnancies                 111
      Glucose                     5
      BloodPressure              35
      SkinThickness             227
      Insulin                   374
      BMI                        11
      DiabetesPedigreeFunction    0
      Age                         0
      Outcome                     0
      dtype: int64
```

Brakujące wartości cech były w danych oznaczone zerami (nie jest to prawidłowe oznaczenie ponieważ np. poziom glukozy nie może być na poziomie zerowym) dlatego zmieniliśmy ich oznaczenie na NaN. Dzięki temu wiemy w których kolumnach i jak duże te braki danych występują. Najwięcej brakujących danych mamy w kolumnie Insulin i SkinThickness.

FUNKCJA KTÓRA MA ZA ZADANIE ZASTĄPIĆ WARTOŚCI ZEROWE MEDIANĄ

```
[13]: def deal_with_Nan(column):
          temp = df[df[column].notnull()]
          temp = temp[[column, 'Outcome']].groupby(['Outcome'])[[column]].median().reset_index()
          return temp
```

```
[14]: columns = df.iloc[:,:-1]
      columns
```

ZASTOSOWANIE FUNKCJI

```
[15]: for column in columns: # dealing with missing values
          deal_with_Nan(column)
          df.loc[(df['Outcome'] == 0 ) & (df[column].isna()), column] = deal_with_Nan(column)[column][0]
          df.loc[(df['Outcome'] == 1 ) &  (df[column].isna()),column] = deal_with_Nan(column)[column][1]
```

```
[16]: df.isna().sum()
```

```
[16]: Pregnancies                 0
      Glucose                     0
      BloodPressure               0
      SkinThickness               0
      Insulin                     0
      BMI                         0
      DiabetesPedigreeFunction    0
      Age                         0
      Outcome                     0
      dtype: int64
```

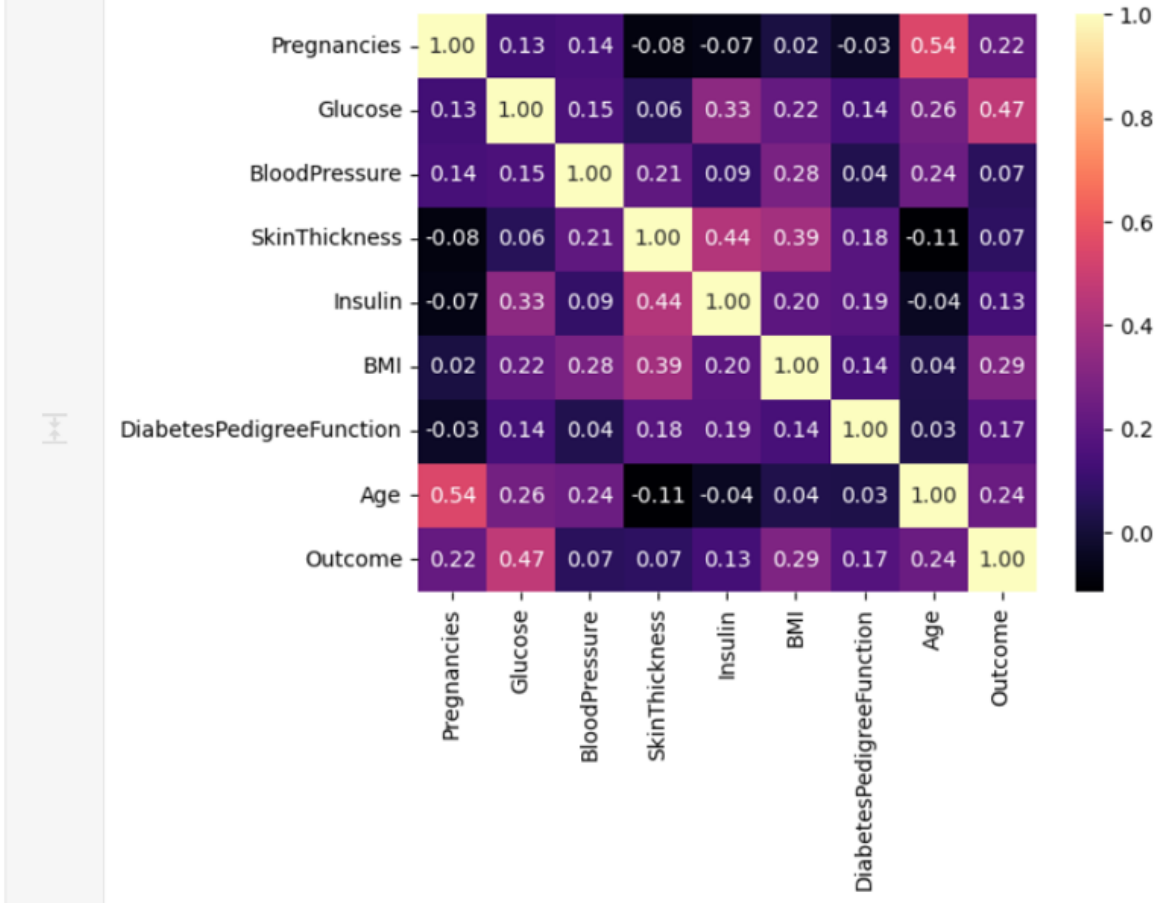
Widzimy że wszystkie brakujące dane zostały zastąpione przez mediany tychże zmiennych i nie mamy już brakujących wartości w naszych danych.

1.2.2 Skorelowanie Danych

Poniżej znajduje się mapa cieplna która ma za zadanie pokazać korelacje pomiędzy poszczególnymi zmiennymi

```
[9]: sns.heatmap(df.corr(),annot=True, fmt = '.2f', cmap='magma')
```

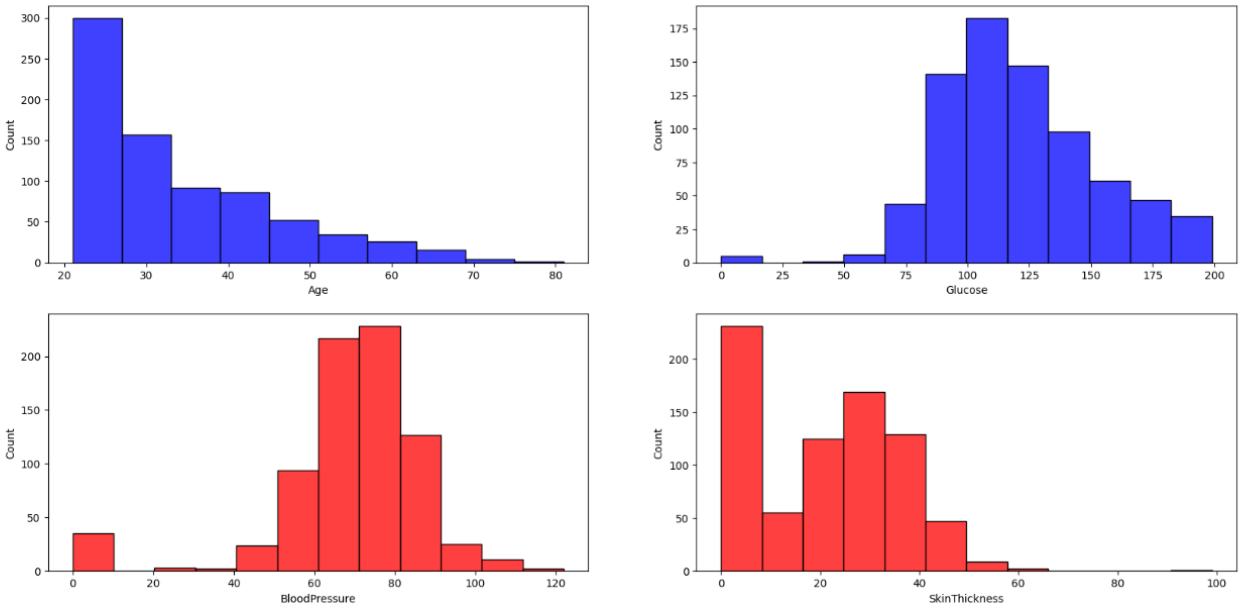
```
[9]: <Axes: >
```



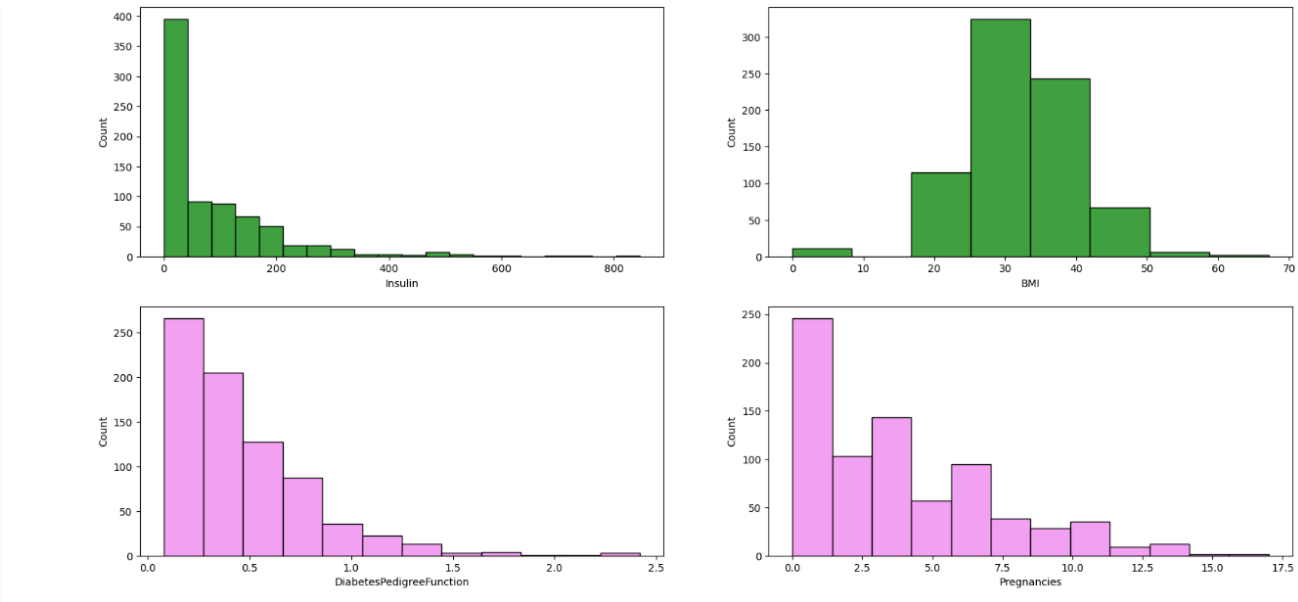
Widzimy że najbardziej skorelowanymi zmiennymi są poziom Glukozy we krwi i Outcome (0.47) czyli czy osoba jest chora na cukrzyce co może nam mówić że ta zmienna ma duży wpływ na to czy osoba jest chora czy nie oraz Wiek i Liczba Ciąż(0.54) co jest zależnością która prawdopodobnie nie wpływa na nasze dane.

1.2.4 Rozkłady zmiennych

Poniżej znajdują się histogramy rozkładu zmiennych: Wieku, Poziomu Glukozy, Ciśnienia Krwi i Grubości Skóry



Poniżej znajdują się histogramy rozkładu zmiennych: Poziomu Insuliny, BMI, Wskaźnika dziedziczenia cukrzycy i Liczby Ciąż



1.2.5 Wartości Odstające

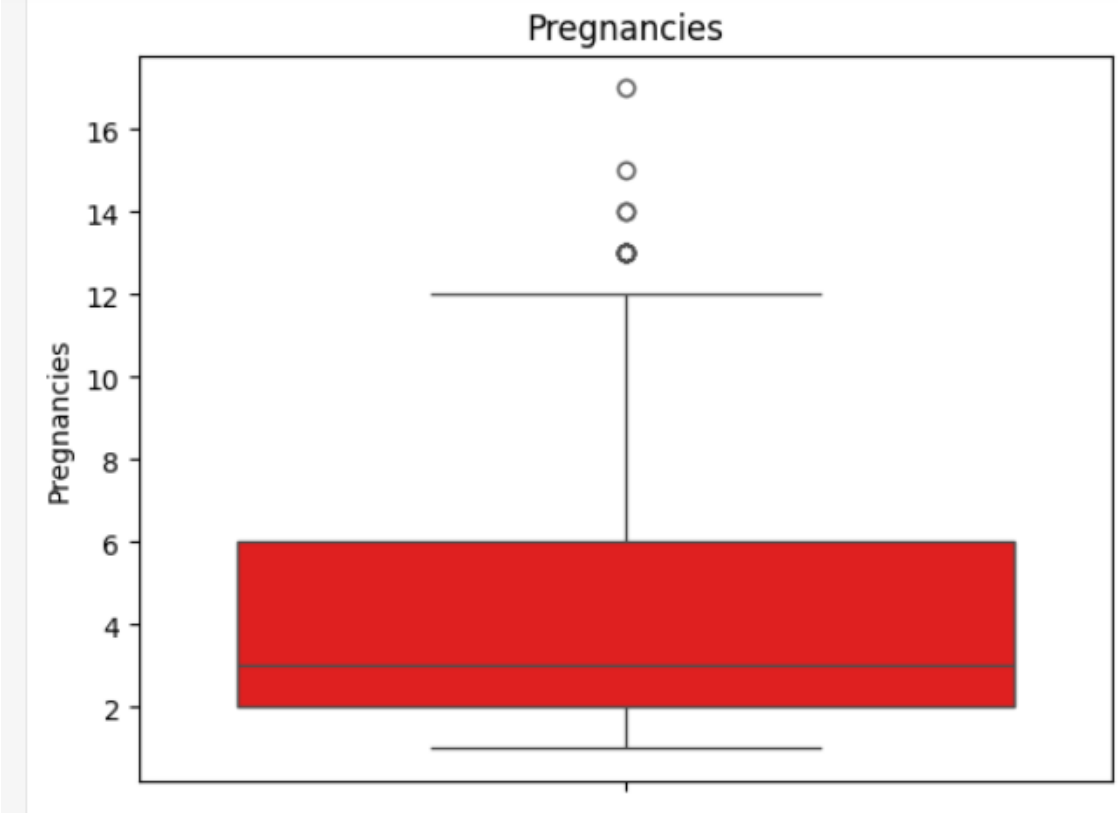
Wykonaliśmy wykresy pudełkowe (box plot) dla wszystkich cech, w których zidentyfikowaliśmy wartości odstające za pomocą poniższej funkcji.

[50]:
```
for column in df.columns: # function for detecting outliers
    Q3 = df[column].quantile(0.75)
    Q1 = df[column].quantile(0.25)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5*IQR
    upper_bound = Q3 + 1.5*IQR
    if df[(df[column] > upper_bound)].any(axis=None):
        print(column, "yes")
    else:
        print(column, "no")
```

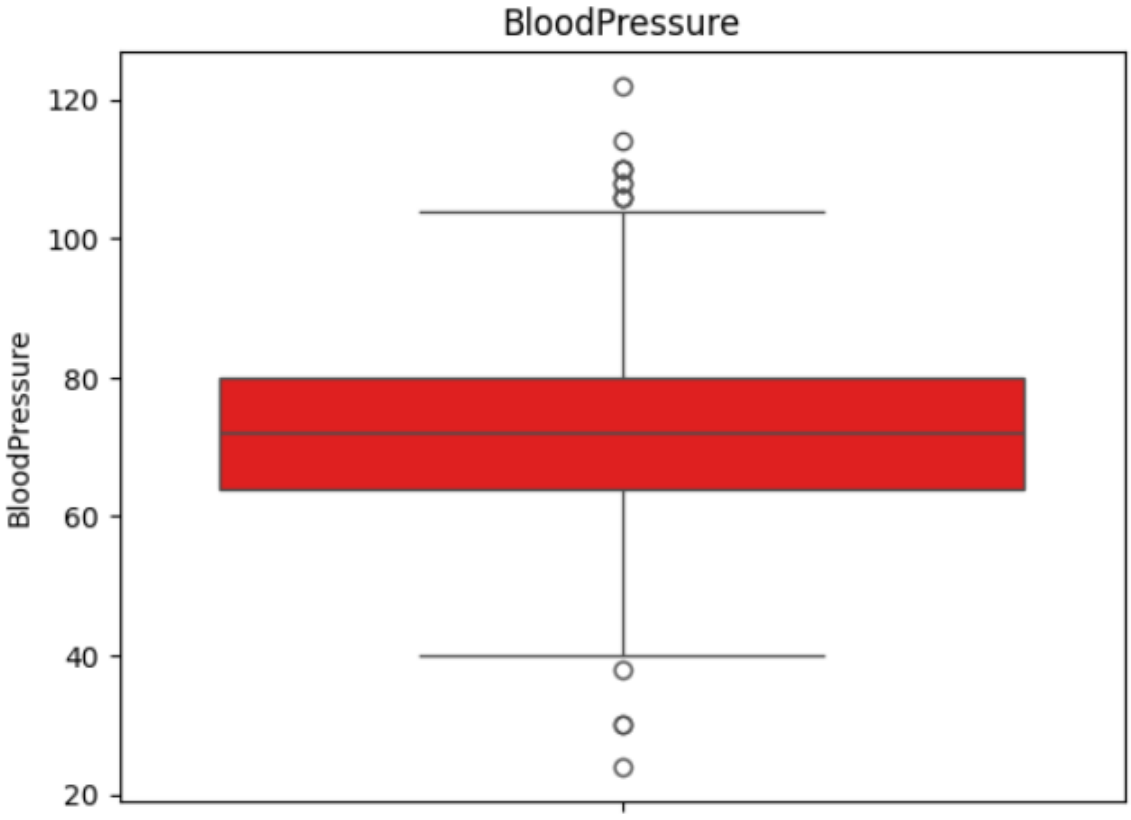
```
Pregnancies yes
Glucose no
BloodPressure yes
SkinThickness yes
Insulin yes
BMI yes
DiabetesPedigreeFunction yes
Age yes
Outcome no
```

Poniżej znajdują się wykresy pudełko-wąsy dla zmiennych Pregnancies, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age

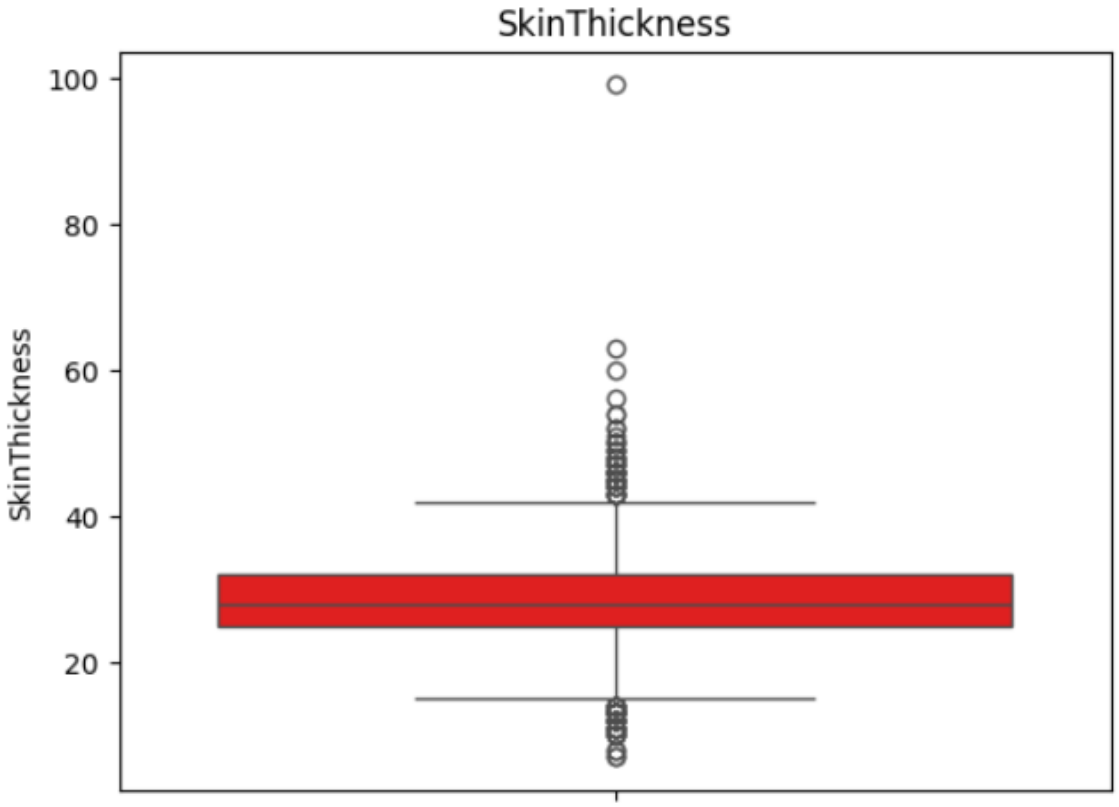
PREGNANCIES:



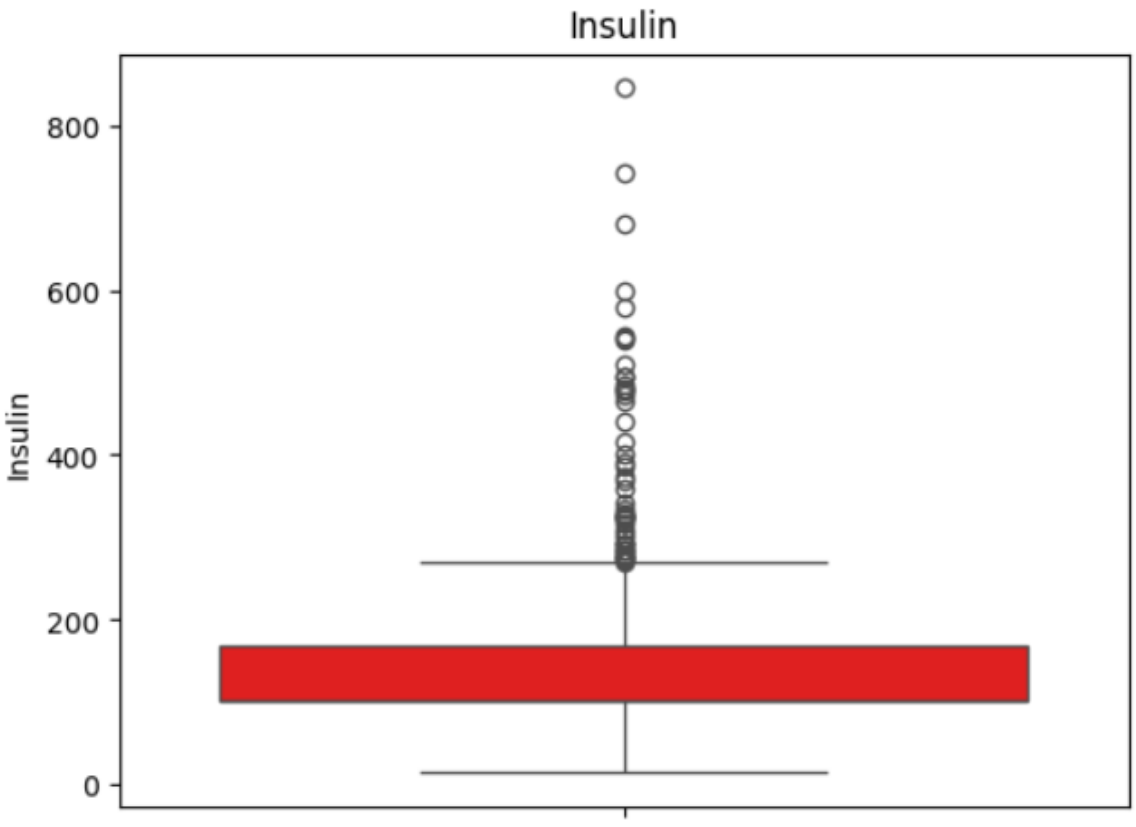
BLOOD PRESSURE:



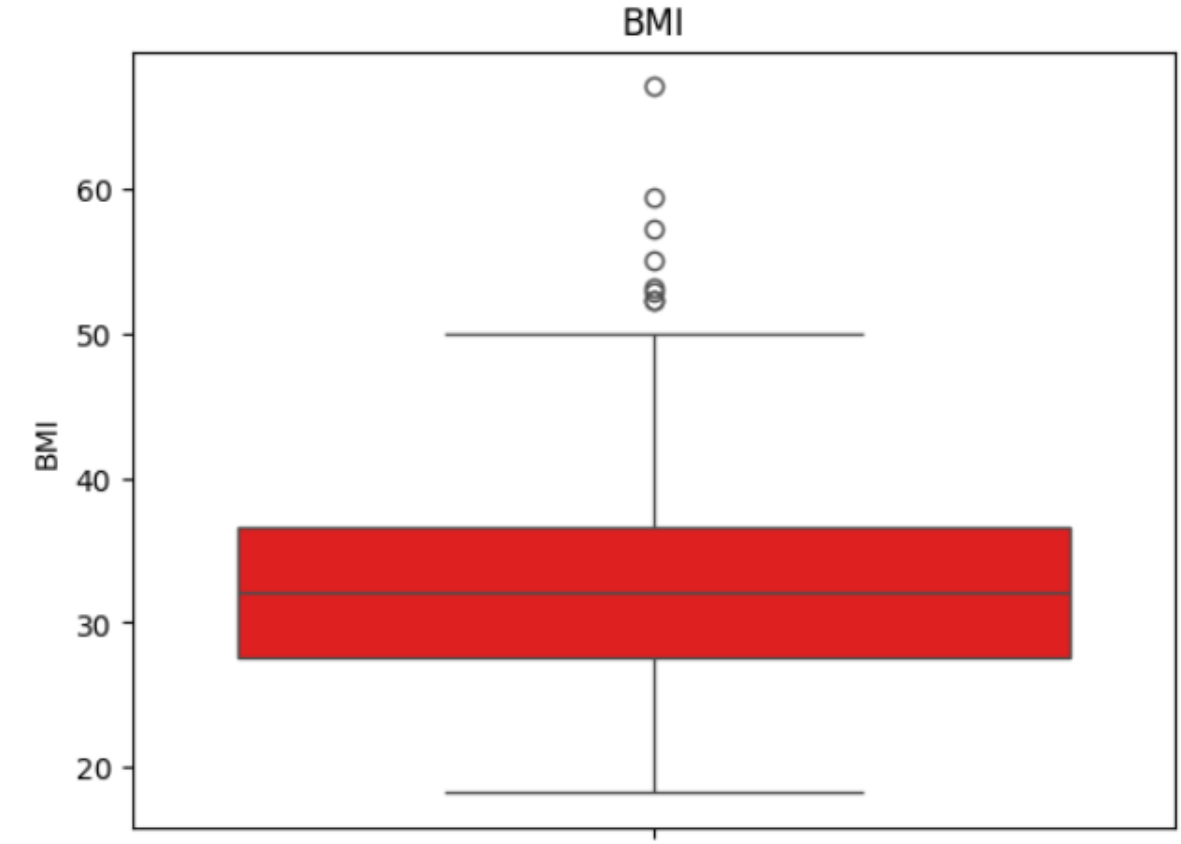
SKIN THICKNESS:



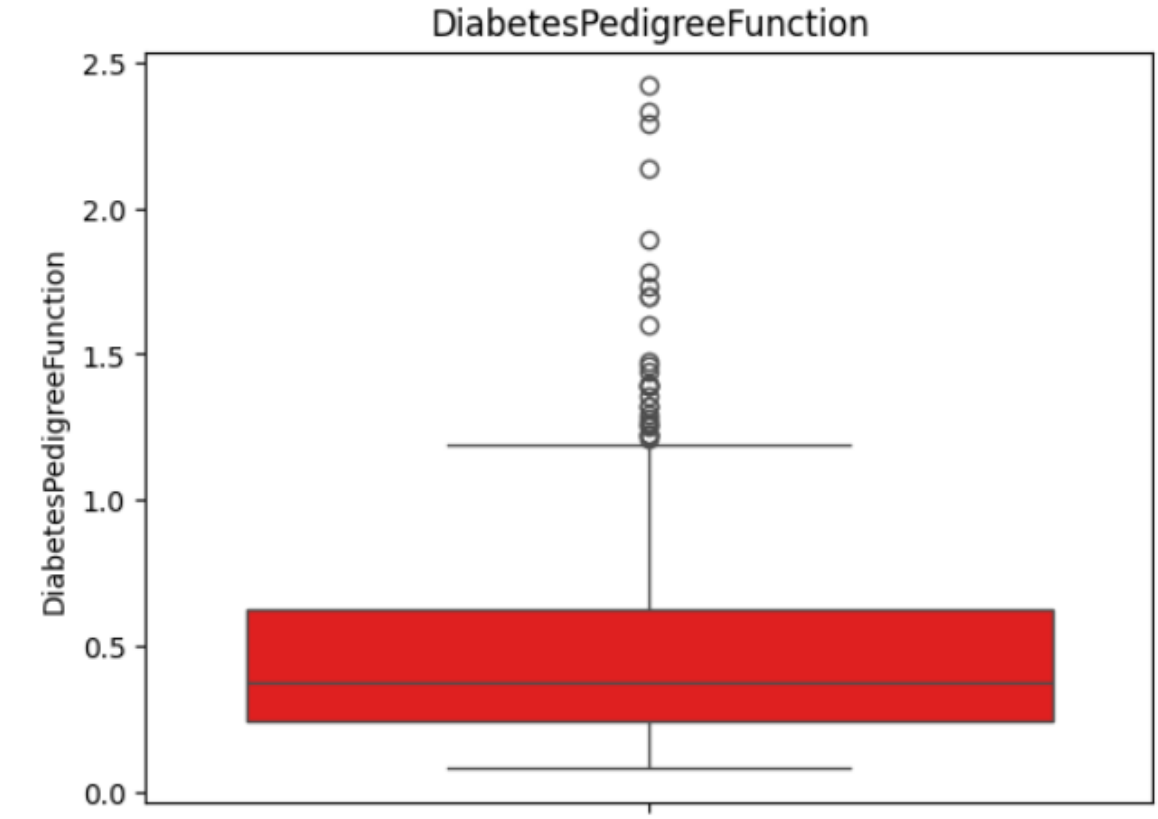
INSULIN:



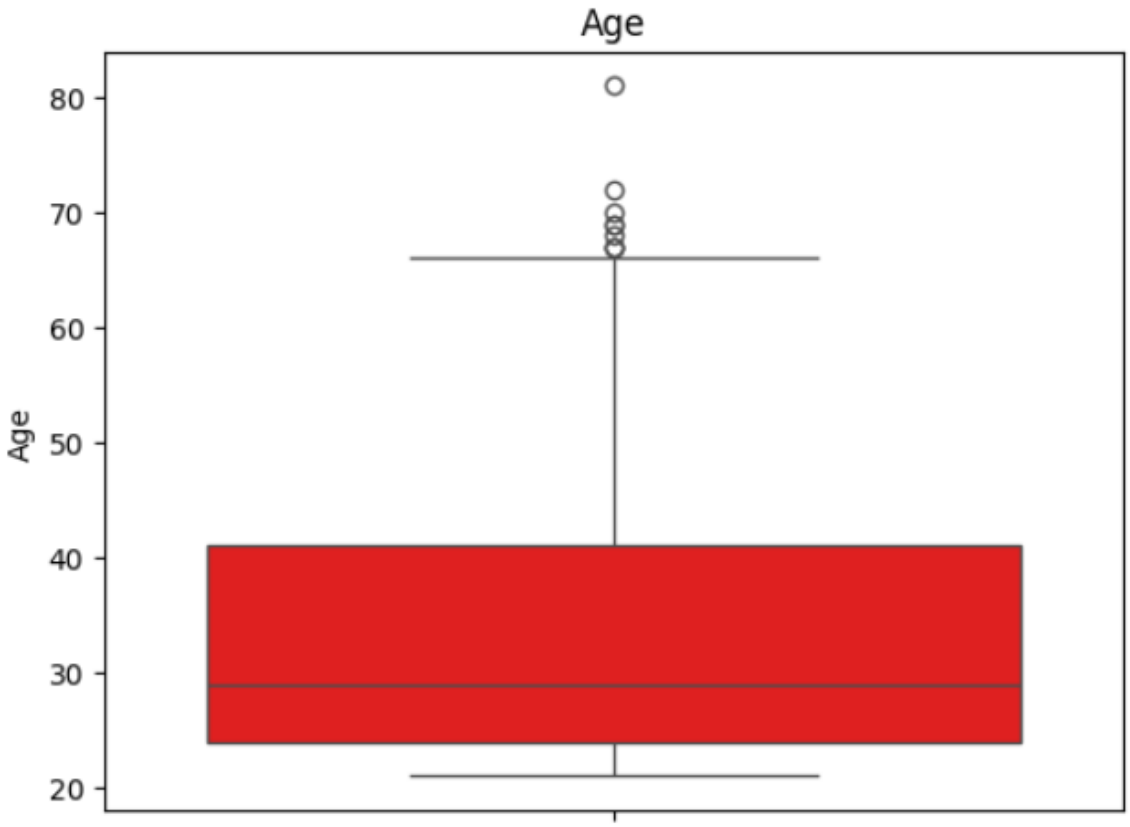
BMI:



DIABETES PEDIGREE FUNCTION:



AGE:



Widzimy że najwięcej wartości odstających mamy w kolumnach SkinThickness, Insulin, DiabetesPedigreeFunction. Poniżej znajduję się funkcja która ma za zadanie usunąć wartości odstające.

```python
def clean_outliers(data,column): # function for clearing outliers
    Q3 = df[column].quantile(0.75)
    Q1 = df[column].quantile(0.25)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5*IQR
    upper_bound = Q3 + 1.5*IQR
    lower_filter = lower_bound < data[column]
    upper_filter = upper_bound > data[column]
    cleaned_data = lower_filter & upper_filter
    data = data[cleaned_data]
    ##print("{} of dataset after column {}".format(data.shape, columns))
    return data
```

```python
df.shape
```

```
(768, 9)
```

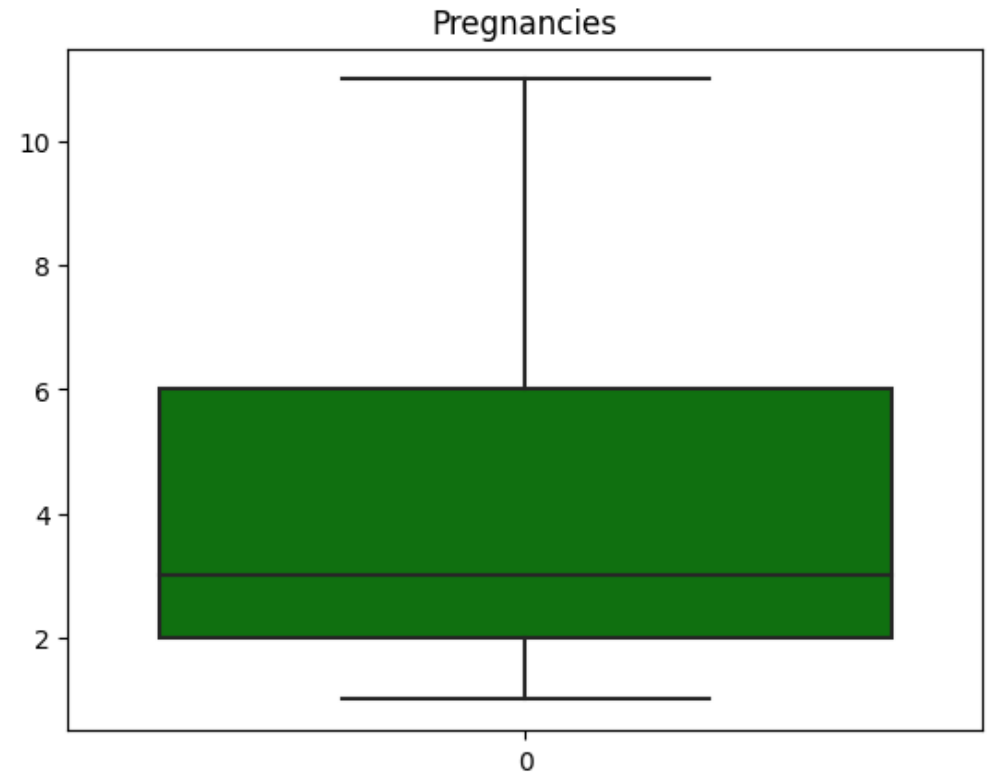
```python
df_clean = df.copy() # clearing outliers (deleting them)
for c in df.columns:
    df_clean = clean_outliers(df_clean,c)
df_clean.shape
```

```
(573, 9)
```

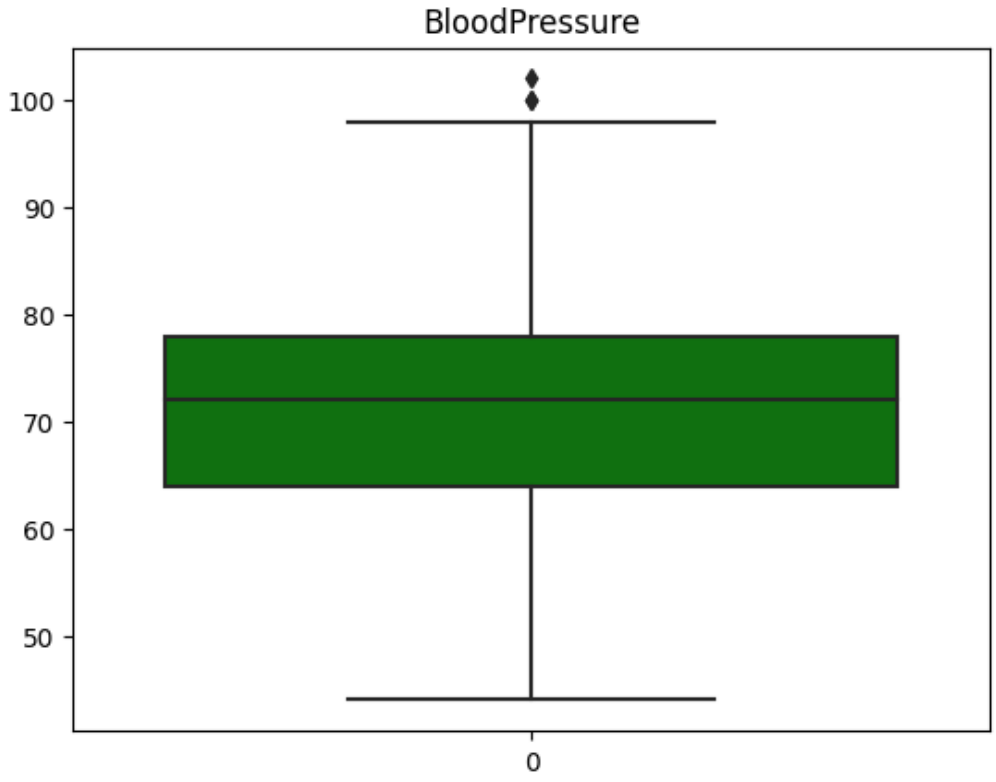
Widzimy że stosując tą funkcje na naszych kolumnach pozbyliśmy się 195 wierszy.

Poniżej znajdują się wykresy po uprzątnięciu danych w kolumnach:

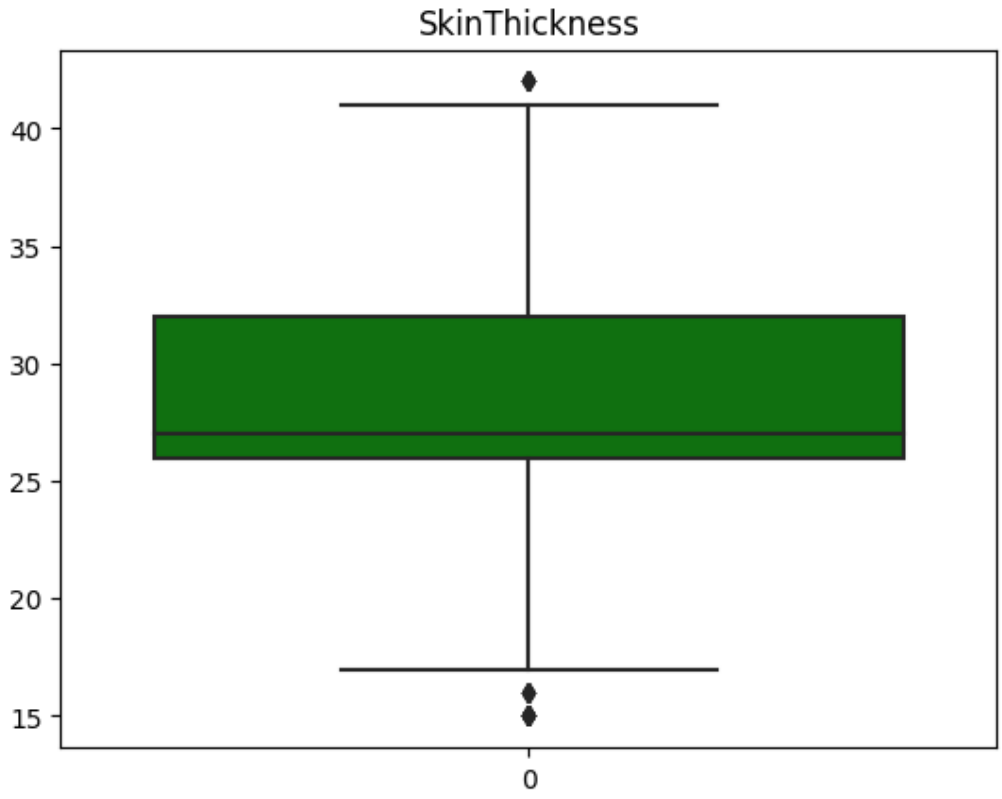
Pregnancies:



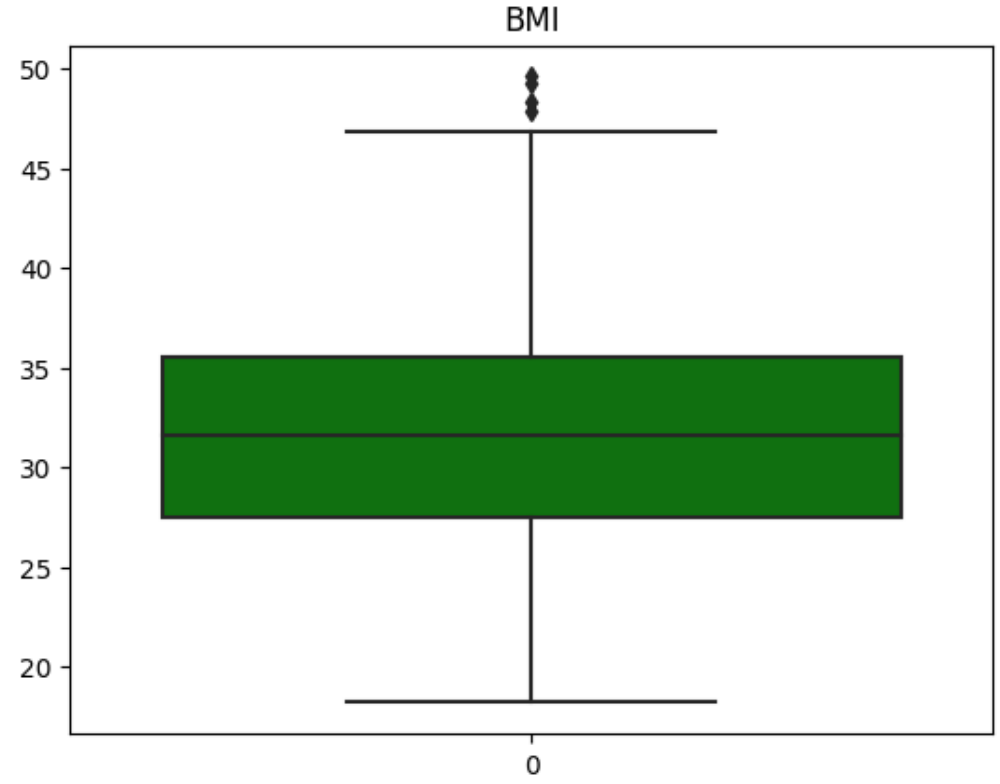
BloodPressure:



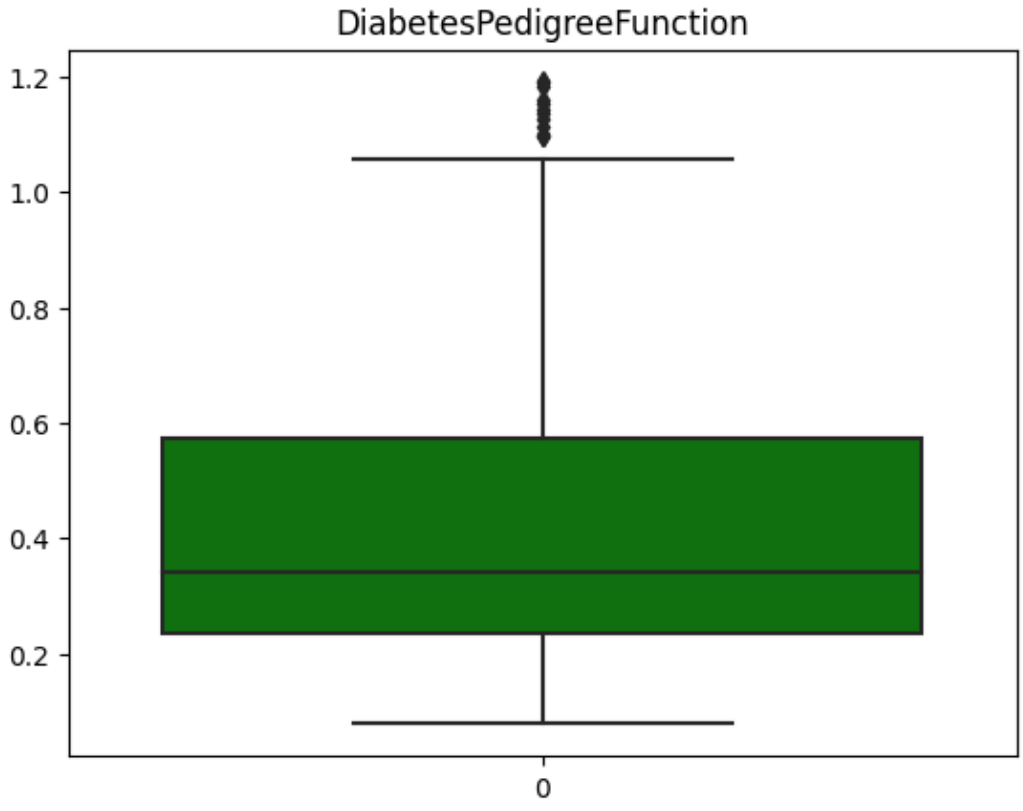
SkinThickness:



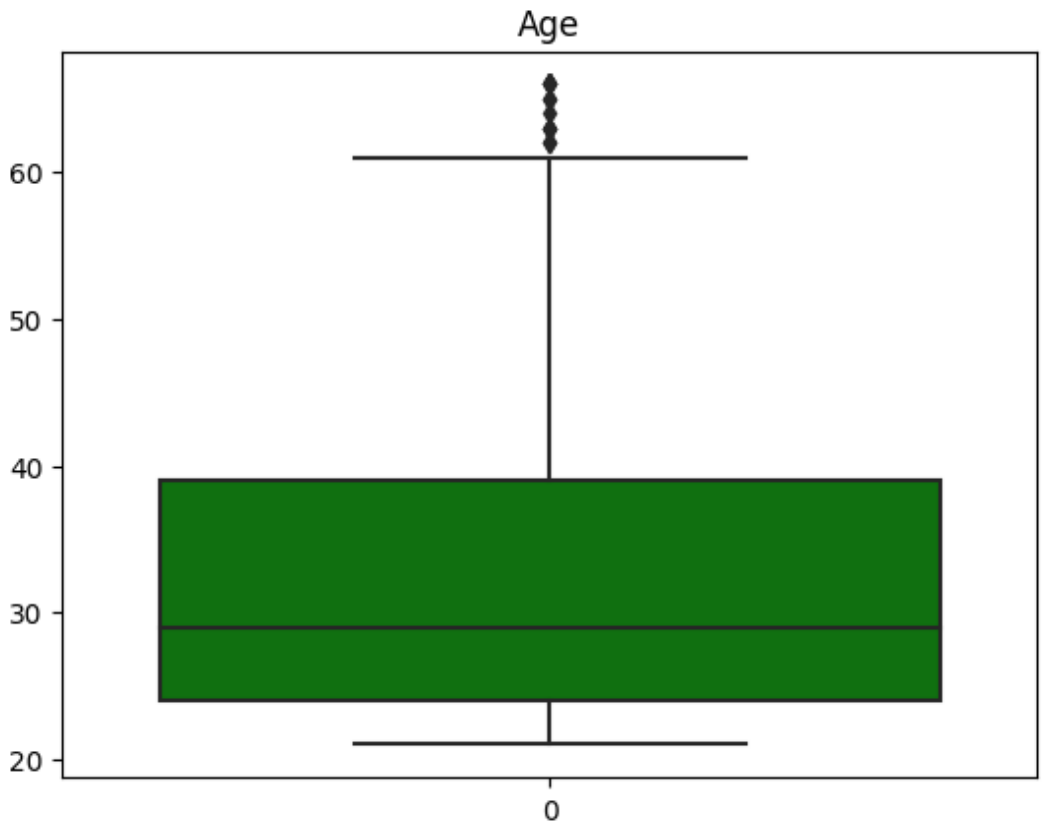
BMI:



DiabetesPedigreeFunction:



Age:



1.3 Tworzenie Nowych Zmiennych

Aby polepszyć funkcjonowanie modelu tworzymy 3 nowe cech, które będą miały charakter zmiennej kategorycznej a następnie stosujemy metode OneHotEncoding.

OneHotEncoding jest techniką przekształcania zmiennych kategorycznych w format, który może być wykorzystywany przez modele uczenia maszynowego. W tej metodzie każda kategoria cechy kategorycznej jest przekształcana na oddzielną zmienną, a wartość tej zmiennej jest ustawiana na 1 lub 0, wskazując obecność lub brak danej kategorii.

TWORZENIE ZMIENNEJ KATEGORYCZNEJ BMI

```
]:
NewBMI = ["Underweight","Normal weight","Overweight","Obesity Type 1","Obesity Type 2","Obesity Type 3"]
df_tree.loc[df["BMI"] < 18.5, "BMI_cat"] = NewBMI[0]
df_tree.loc[(df["BMI"] >= 18.5) & (df["BMI"] <= 24.9), "BMI_cat"] = NewBMI[1]
df_tree.loc[(df["BMI"] >= 25) & (df["BMI"] <= 29.9), "BMI_cat"] = NewBMI[2]
df_tree.loc[(df["BMI"] >= 30) & (df["BMI"] <= 34.9), "BMI_cat"] = NewBMI[3]
df_tree.loc[(df["BMI"] >= 35) & (df["BMI"] <= 39.9), "BMI_cat"] = NewBMI[4]
df_tree.loc[df["BMI"] >= 40, "BMI_cat"] = NewBMI[5]
df_tree.head()
```

TWORZENIE ZMIENNEJ KATEGORYCZNEJ GLUKOZY

```
NewGlucose = ["Low", "Normal", "Elevated", "High"]
df_tree.loc[df["Glucose"] < 70, "Glucose_cat"] = NewGlucose[0]
df_tree.loc[(df["Glucose"] >= 70) & (df["Glucose"] < 100), "Glucose_cat"] = NewGlucose[1]
df_tree.loc[(df["Glucose"] >= 100) & (df["Glucose"] < 126),"Glucose_cat"] = NewGlucose[2]
df_tree.loc[df["Glucose"] >= 126, "Glucose_cat"] = NewGlucose[3]
df_tree.head()
```

TWORZENIE ZMIENNEJ KATEGORYCZNEJ INSULINY

```
NewInsulin = ["Normal","Abnormal"]
df_tree.loc[(df["Insulin"] < 166.0) & (df["Insulin"] > 16.0),"Insulin_cat"] = NewInsulin[0]
df_tree.loc[(df["Insulin"] >= 166.0) | (df["Insulin"] <= 16.0),"Insulin_cat"] = NewInsulin[1]
df_tree.head()
```

WYGLĄD RAMKI DANYCH PO DODANIU NOWYCH CECH KATEGORYCZNYCH

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome	BMI_cat	Glucose_cat	Insulin_cat
0	6.0	148.0	72.0	35.0	169.5	33.6	0.627	50	1	Obesity Type 1	High	Abnormal
1	1.0	85.0	66.0	29.0	102.5	26.6	0.351	31	0	Overweight	Normal	Normal
2	8.0	183.0	64.0	32.0	169.5	23.3	0.672	32	1	Normal weight	High	Abnormal
3	1.0	89.0	66.0	23.0	94.0	28.1	0.167	21	0	Overweight	Normal	Normal
5	5.0	116.0	74.0	27.0	102.5	25.6	0.201	30	0	Overweight	Elevated	Normal

WYGLĄD RAMKI DANYCH PO ZASTOSOWANIU METODY ONE HOT ENCODING

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome	BMI_cat_Obesity Type 1	BMI_cat_Obesity Type 2
0	6	148	72	35	169	33	0	50	1	1	0
1	1	85	66	29	102	26	0	31	0	0	0
2	8	183	64	32	169	23	0	32	1	0	0
3	1	89	66	23	94	28	0	21	0	0	0
5	5	116	74	27	102	25	0	30	0	0	0

BMI_cat_Overweight	BMI_cat_Underweight	Insulin_cat_Normal	Glucose_cat_High	Glucose_cat_Low	Glucose_cat_Normal
0	0	0	1	0	0
1	0	1	0	0	1
0	0	0	1	0	0
1	0	1	0	0	1
1	0	1	0	0	0

2 Cel Projektu

Naszym celem projektu jest zbudowanie modelu który będzie przewidywał(klasyfikował) czy pacjent choruje na cukrzyce(zmienna binarna Outcome) na podstawie dostarczonych danych zdrowotnych tj. ciśnienia krwi, poziomu glukozy, poziomu

insuliny, wskaźnika dziedziczenia cukrzycy, wieku, liczby ciąż i BMI i nowych cech kategorycznych.

3 Wybór Metody

Wybraliśmy model Drzewa Decyzyjnego

Drzewo decyzyjne to algorytm uczenia maszynowego używany zarówno do zadań klasyfikacyjnych, jak i regresyjnych. Model działa poprzez dzielenie danych wejściowych na mniejsze podzbiory na podstawie pewnych kryteriów, takich jak entropia, współczynnik Gini'ego lub zysk informacji. Każdy węzeł drzewa reprezentuje zmienną wejściową (cechę), każda gałąź przedstawia warunek podziału (na podstawie wartości cechy), a każdy liść reprezentuje klasę lub wartość docelową (dla klasyfikacji lub regresji). Przy podziale drzewa decyzyjnego, oblicza się wartość kryterium podziału dla każdej cechy i wybiera tę, która maksymalizuje (dla zysku informacji) lub minimalizuje (dla współczynnika Gini) kryterium podziału.

Proces budowy drzewa decyzyjnego

1. **Wybór najlepszego podziału**: Dla każdego węzła drzewa wybierany jest najlepszy podział danych na podstawie kryterium jakości podziału (np. entropia, współczynnik Gini).
2. **Podział danych**: Dla wybranej cechy dzieli się dane na mniejsze podzbiory.
3. **Rekurencja**: Proces jest powtarzany rekurencyjnie dla każdego nowego podzbioru danych, tworząc kolejne węzły i gałęzie, aż do spełnienia kryterium zatrzymania (np. osiągnięcie maksymalnej głębokości drzewa, minimalna liczba próbek w węźle)

Uzasadnienie Wyboru:

- **Intuicyjność i łatwość interpretacji**:
    - Drzewa decyzyjne są jednymi z najbardziej intuicyjnych i łatwych do zrozumienia modeli. Wizualizacja drzewa decyzyjnego pozwala zrozumieć, jak model podejmuje decyzje, co jest niezwykle istotne w kontekście medycznym, gdzie interpretowalność modelu może być kluczowa.
- **Obsługa zmiennych kategorycznych i numerycznych**:
    - Drzewa decyzyjne radzą sobie zarówno ze zmiennymi kategorycznymi, jak i numerycznymi, co jest korzystne przy pracy z różnorodnymi danymi medycznymi, które mogą obejmować zarówno zmienne liczbowe (np. wyniki badań krwi), jak i kategoryczne (np. płeć, typ choroby).

Elastyczność i adaptacyjność:

- Drzewa decyzyjne są bardzo elastyczne i mogą modelować złożone relacje między cechami a wynikami. Mogą także łatwo uwzględniać interakcje między różnymi

cechami, co jest ważne w kontekście medycznym, gdzie często występują skomplikowane interakcje między różnymi zmiennymi zdrowotnymi.

Szybkość i efektywność:

- Drzewa decyzyjne są relatywnie szybkie do trenowania i predykcji, co jest korzystne przy pracy z dużymi zbiorami danych medycznych.

Podsumowując, drzewa decyzyjne oferują wiele zalet, które czynią je odpowiednim wyborem do problemu klasyfikacji pacjentów pod kątem występowania choroby, zwłaszcza w kontekście danych medycznych, gdzie interpretowalność, elastyczność i efektywność są kluczowe.

4 Zastosowanie danej metody do danych

4.1 Podział na zbiór uczący i testowy

Definiowanie tego którą cechę będziemy przewidywać(Outcome) czyli czy osoba jest chora czy nie. Dzielenie danych na część testową uczącą w stosunku 80% - ucząca i 20% - testowa

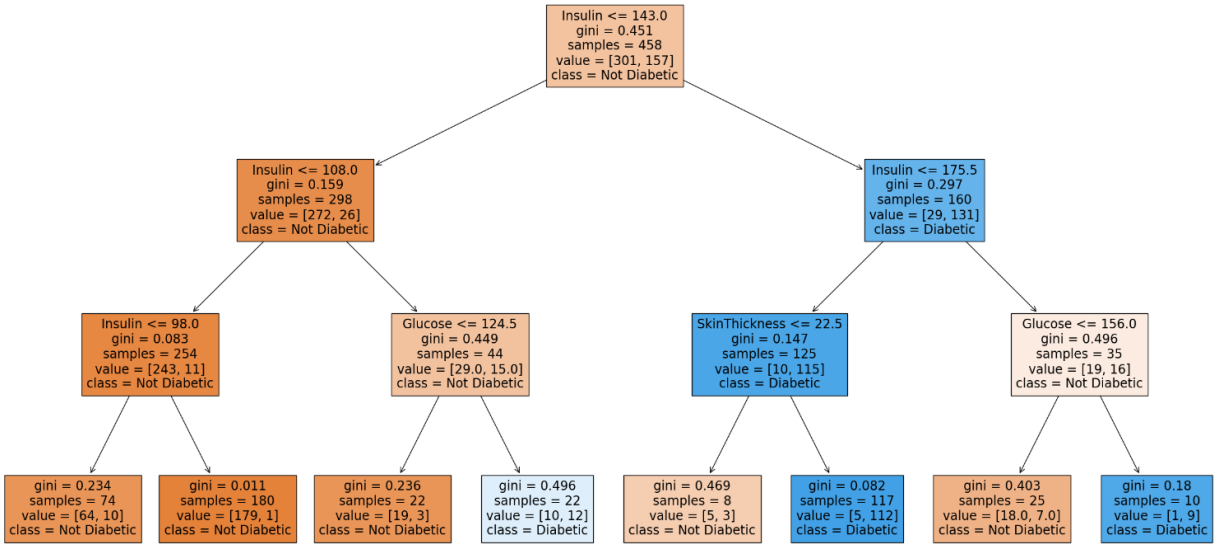
```
X_t = df_tree_encoded.drop('Outcome', axis = 1) # defining what I will be predicting ( the Outcome value)
Y_t = df_tree_encoded['Outcome']
X_train, X_test, Y_train, Y_test = train_test_split(X_t, Y_t, test_size = 0.2, random_state = 45) # spliting dataframe into
train and test part
```

4.2 Definiowanie modelu

```
from sklearn.tree import DecisionTreeClassifier, plot_tree
DecisionTree = DecisionTreeClassifier(max_depth = 3)
DecisionTree.fit(X_train,Y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3)
```

Wykres Drzewa Decyzyjnego:



4.3 Walidacja modelu

W walidacji modelu skupimy się głównie na wskaźniku Accuracy Score(Mierzy, jak często model poprawnie klasyfikuje dane) który chcemy żeby był jak największy. Wykonamy walidacje krzyżową, macierz pomyłek jak i również sprawdzimy wskaźniki takie jak F1 Score, Recall Score i AUC Score.

4.3.1 Accuracy Score, F1 Score, Recall Score i AUC Score

```
print(f"Accuracy score is {accuracy_score(Y_test, DecisionTree.predict(X_test))}")
print(f"F1 score is {f1_score(Y_test, DecisionTree.predict(X_test))}")
print(f"Recall score is {recall_score(Y_test, DecisionTree.predict(X_test))}")
print(f"AUC score is {roc_auc_score(Y_test, DecisionTree.predict(X_test))}")
```

```
Accuracy score is 0.9130434782608695
F1 score is 0.8333333333333334
Recall score is 0.8928571428571429
AUC score is 0.9061986863711001
```

Wskaźnik dokładności (Accuracy score): Wskaźnik ten mierzy ogólną skuteczność modelu poprzez liczbę poprawnych klasyfikacji podzieloną przez liczbę wszystkich klasyfikacji. Wysoki wynik wskaźnika dokładności oznacza, że model poprawnie klasyfikuje większość próbek. Wynik wynoszący 0.913 oznacza, że 91.3% wszystkich klasyfikacji było poprawnych. Innymi słowy, zastosowany model klasyfikacyjny poprawnie sklasyfikował 91.3% wszystkich próbek.

Wskaźnik F1 (F1 score): Jest to średnia harmoniczna precyzji i odwołania. Precyzja mierzy stopień, w jakim model wykrywa prawdziwie pozytywne przypadki, podczas gdy odwołanie mierzy zdolność modelu do wykrycia wszystkich rzeczywiście pozytywnych przypadków. Wysoki wskaźnik F1 oznacza, że model osiąga równowagę między precyzją a odwołaniem. Wartość wynosząca 0.833 oznacza, że model osiągnął dobrą równowagę między precyzją (dokładność pozytywnych przewidywań) a odwołaniem (odnalezienie

wszystkich rzeczywiście pozytywnych przypadków). Jest to szczególnie istotne w przypadku niezbalansowanych danych.

Wskaźnik odwołania (Recall score): Mierzy on zdolność modelu do wykrycia wszystkich rzeczywiście pozytywnych przypadków. Wyższy wskaźnik odwołania oznacza, że model lepiej identyfikuje rzeczywiście pozytywne przypadki, co jest istotne w przypadku problemów, w których brak detekcji pozytywnych przypadków może być kosztowny. Wynik 0.893 oznacza, że 89.3% rzeczywiście pozytywnych przypadków zostało zidentyfikowanych poprawnie przez model. W kontekście klasyfikacji problemu medycznego, wyższy wskaźnik odwołania oznacza lepszą zdolność modelu do wykrywania rzeczywiście pozytywnych przypadków.

AUC score (Obszar pod krzywą ROC): Ten wskaźnik mierzy zdolność modelu do rozróżniania między klasami poprzez ocenę obszaru pod krzywą charakterystyki pracy odbiornika (ROC). Im wyższy jest obszar pod krzywą ROC, tym lepiej model rozróżnia między klasami. Wynik 0.906 oznacza, że model ma dobrą zdolność do rozróżniania między klasami. Im wyższy jest ten wskaźnik, tym lepiej model rozróżnia między pozytywnymi i negatywnymi przypadkami.

4.3.2 Walidacja Krzyżowa

Wskaźniki walidacji krzyżowej (Cross-validation scores): Oceniają one skuteczność modelu na nowych danych poprzez testowanie go na różnych podzbiorach danych treningowych i walidacyjnych. Średnia wartość cross-walidacji i mediana są miarami stabilności i generalizacji modelu, ponieważ potwierdzają, czy model zachowuje swoją wydajność na różnych zestawach danych walidacyjnych. Średnia wartość cross-walidacji wynosząca 0.897 sugeruje, że model generalizuje dobrze na nowe dane spoza zbioru treningowego. Mediana wyników cross-walidacji również wynosząca 0.897 potwierdza spójność modelu na różnych podzbiorach danych walidacyjnych. Jest to ważne dla oceny stabilności i skuteczności modelu na różnych zestawach danych testowych.

```
from sklearn.model_selection import cross_val_score, KFold
kfold = KFold(n_splits=15, shuffle=True, random_state=42)
cv_results = cross_val_score(DecisionTree, X_t, Y_t, cv=kfold, scoring='accuracy')
cv_results
```

```
array([0.8974359 , 0.84615385, 0.92307692, 0.97368421, 0.89473684,
       0.97368421, 0.78947368, 0.86842105, 0.94736842, 0.78947368,
       0.89473684, 0.92105263, 0.92105263, 0.89473684, 0.92105263])
```

```
print(f"Mean cross-validation score:{np.mean(cv_results)}")
print(f"Median of cross-validation scores:{np.median(cv_results)}")
```

```
Mean cross-validation score:0.8970760233918128
Median of cross-validation scores:0.8974358974358975
```

4.3.3 Macierz Pomyłek

```
conf_matrix = confusion_matrix(Y_test, Y_prediction)
conf_matrix
```

```
array([[75, 12],
       [ 6, 22]], dtype=int64)
```

W macierzy pomyłek odnotowaliśmy 75 przypadków poprawnie sklasyfikowanych wyników negatywnych oraz 22 przypadki poprawnie sklasyfikowanych wyników pozytywnych. Wystąpiło także 12 przypadków niepoprawnie sklasyfikowanych wyników pozytywnych oraz 6 przypadków niepoprawnie sklasyfikowanych wyników negatywnych.

5 Podsumowania i Wnioski

Analiza wyników walidacji modelu drzewa decyzyjnego dostarczyła istotnych informacji na temat jego skuteczności i generalizacji. Wartość wskaźnika dokładności (Accuracy score) wynosząca 0.913 oznacza, że model poprawnie sklasyfikował 91.3% wszystkich próbek, co wskazuje na dobrą ogólną skuteczność modelu. Wskaźnik F1 (F1 score) wynoszący 0.833 potwierdza, że model osiągnął dobrą równowagę między precyzją a odwołaniem, co jest istotne szczególnie w przypadku niezbalansowanych danych. Wskaźnik odwołania (Recall score) na poziomie 0.893 świadczy o wysokiej zdolności modelu do wykrywania rzeczywiście pozytywnych przypadków, co jest kluczowe w kontekście problemów medycznych, gdzie brak detekcji pozytywnych przypadków może mieć poważne konsekwencje. AUC score na poziomie 0.906 potwierdza, że model dobrze rozróżnia między klasami, co jest istotne dla skutecznego identyfikowania zarówno pozytywnych, jak i negatywnych przypadków.

Dodatkowo, wyniki walidacji krzyżowej potwierdzają stabilność i generalizację modelu na nowych danych, co sugeruje, że model zachowuje swoją skuteczność na różnych zestawach danych walidacyjnych.

W analizie macierzy pomyłek zauważamy, że model poprawnie sklasyfikował większość przypadków zarówno dla wyników pozytywnych, jak i negatywnych, ale wystąpiły również pewne błędy, które wymagają dalszej analizy i optymalizacji.

Wnioski:

Model drzewa decyzyjnego osiągnął wysoką skuteczność w klasyfikacji danych zdrowotnych, co sugeruje jego potencjał w diagnozie chorób. Istotne jest zachowanie równowagi między precyzją a odwołaniem, aby model skutecznie wykrywał zarówno pozytywne, jak i negatywne przypadki. Wartości wskaźników walidacji krzyżowej świadczą o stabilności i generalizacji modelu na nowe dane, co potwierdza jego skuteczność w praktyce.

Jednakże, aby model mógł osiągnąć pełny potencjał, konieczne jest dalsze badanie i optymalizacja. W szczególności, analiza błędów popełnionych przez model w macierzy pomyłek może pomóc zidentyfikować obszary wymagające ulepszeń. Dodatkowo, istotne jest

monitorowanie skuteczności modelu w praktyce oraz dostosowywanie go do ewentualnych zmian w danych lub warunkach zewnętrznych.

Wnioski te wskazują na potrzebę kontynuacji prac nad modelem drzewa decyzyjnego oraz jego dalszego doskonalenia, co może przyczynić się do jeszcze lepszych rezultatów w przewidywaniu chorób na podstawie danych zdrowotnych.