Uniwersytet Gdański
Wydział Matematyki, Fizyki i Informatyki

Informatyka Ogólnoakademicka II stopień

Julia Komorowska

Nr albumu: 266386

Specjalność: Ogólna

Rodzaj studiów: Stacjonarne

Testowanie klasyfikatorów na wybranej bazie danych

Gdańsk 2022

Streszczenie

Zakres pracy obejmuje projekt polegający na testowaniu klasyfikatorów i porównaniu ich do siebie. Cała praca została także opisana w pliku:

- projekt.ipynb

Cały projekt został napisany w języku Python w Jupyter Notebook.

Treść zadania

Celem projektu (typu d) jest przetestowanie klasyfikatorów na wybranej bazie danych. Można wybrać następującą bazę danych

- COVID19 `https://www.kaggle.com/datasets/meirnizri/covid19-dataset`

Typ D: Uczenie maszynowe – klasyfikacja lub regresja

Naszym zadaniem jest wzięcie dużej bazy danych z Internetu (lub stworzenie własnej) i przetestowanie, jak działają na niej klasyfikatory poznane na zajęciach i na wykładzie tzn.

- **kNN** (najlepiej dla wielu k)
- **drzewa decyzyj**ne (też w paru wersjach) – dodaj wizualizację drzewa!
- **Naive Bayes**
- **Sieci neuronowe** w różnych konfiguracjach tj. różne topologie, funkcje aktywacji, optimizery, learning rate, techniki na regularyzację, itp.

Można alternatywnie do klasyfikacji wziąć też regresję (liniowa, wielomianowa itp.).
Wskazówki:

- Ciekawe bazy danych możesz znaleźć np. na stronie https://www.kaggle.com/datasets. Nie dość, że mają datasety, to również konkursy z rankingami i fora dyskusyjne. Warto tę stronę przejrzeć. Kilka przykładowych propozycji znajdziesz dalej w sekcji „Propozycje tematów do wyboru".
- Wybierając bazę danych zwróć uwagę na to czy jest odpowiednio duża (mile widziane minimum kilka tysięcy rekordów), interesująca i nieoklepana, oraz czy ma dane umożliwiające klasyfikację (kolumna z klasą).
- W eksperymentach sprawdź klasyfikatory, które najlepiej działają na Twoim datasecie (dokładność, macierz błędu). W przypadku trenowania sieci neuronowych mile widziane są też krzywe uczenia się.
- Przed klasyfikacją możesz dokonać preprocessingu danych (szukanie błędów, brakujących danych, sensowna obróbka). Przykładowe źródła: https://towardsdatascience.com/data-cleaning-with-python-and-pandasdetecting-missing-values-3e9c6ebcf78b lub https://realpython.com/python-data-cleaning-numpy-pandas/
- Można też zrobić analizę statystyczną danych (częstości występowania danych w każdej z kolumn), co pomoże zaznajomić się zdanymi i wykryć ewentualne błędy.
- W przypadku regresji można porównać regresję liniową z wielomianową i zobaczyć, czy wielomianowa działa o wiele lepiej.

Spis treści

1. Wprowadzenie

Projekt został stworzony na podstawie bazy danych Covid-19 `https://www.kaggle.com/datasets/meirnizri/covid19-dataset`.

1.1. Importowanie paczek

Na początku trzeba zaimportować wszystkie paczki potrzebne do uruchomienia programu.

```
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.neural_network import MLPClassifier
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, plot_confusion_matrix,precisio
from sklearn.linear_model import Perceptron
from sklearn import tree
from sklearn.metrics import confusion_matrix,mean_squared_error, r2_score,
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import make_blobs
from sklearn.tree import export_graphviz,export_text
from sklearn.naive_bayes import GaussianNB,BernoulliNB,CategoricalNB
from sklearn.preprocessing import StandardScaler,OrdinalEncoder
from sklearn.model_selection import learning_curve
from mlxtend.plotting import plot_learning_curves
import seaborn as sns
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from six import StringIO
from IPython.display import Image
import pydotplus
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules
import warnings
import json
warnings.filterwarnings("ignore")
```

1.2. Preprocessing

Wstępne przetwarzanie bazy danych w celu zapewnienia większej wydajności jest pierwszym krokiem do przygotowania naszej bazy danych do użytku.

- Pobranie bazy danych i pokazanie jej kolumn.

```
df = pd.read_csv("covid_data.csv")
for col in df.columns:
    print(col)
```

```
USMER
MEDICAL_UNIT
SEX
PATIENT_TYPE
DATE_DIED
INTUBED
PNEUMONIA
AGE
PREGNANT
DIABETES
COPD
ASTHMA
INMSUPR
HIPERTENSION
OTHER_DISEASE
CARDIOVASCULAR
OBESITY
RENAL_CHRONIC
TOBACCO
CLASIFFICATION_FINAL
ICU
```

- Kolumny:
  - USMER - wskazuje czy pacjent był leczony przez jednostki medyczne pierwszego, drugiego czy trzeciego poziomu.
  - MEDICAL UNIT - jednostka medyczna
  - SEX - 1 to kobieta, 2 to mężczyzna
  - PATIENT TYPE - 1 wrócił do domu, 2 hospitalizowany
  - DATE DIED - data śmierci
  - INTUBED - zaintubowany
  - PNEUMONIA - zapalenie płuc

  - AGE - wiek
  - PREGNANT - ciąża
  - DIABETES - cukrzyca
  - COPD - chroniczne obturacyjne choroby ukladu oddechowego
  - ASTHMA - astma
  - INMSUPR - obniżona odporność
  - HIPERTENSION - nadciśnienie
  - OTHER DISEASE - inne choroby
  - CARDIOVASCULAR - problemy układu sercowo-naczyniowego
  - OBESITY - otyłość
  - RENAL CHRONIC - przewlekła niewydolność nerek
  - TOBACCO - palenie papierosów
  - CLASIFFICATION FINAL - czy dany człowiek ma covida
  - ICU - czy przydzielony do oddziału intensywnej terapii
- Wczytanie pięciu pierwszych i ostatnich wierszy

```
df.head()
```

	USMER	MEDICAL_UNIT	SEX	PATIENT_TYPE	DEATH	INTUBED	PNEUMONIA	AGE	PREGNANT	DIABETES	...	ASTHMA	INMSUPR	HIPERTENSION
0	2	1	1	1	03/05/2020	97	1	65	2	2	...	2	2	1
1	2	1	2	1	03/06/2020	97	1	72	97	2	...	2	2	1
2	2	1	2	2	09/06/2020	1	2	55	97	1	...	2	2	2
3	2	1	1	1	12/06/2020	97	2	53	2	2	...	2	2	2
4	2	1	2	1	21/06/2020	97	2	68	97	1	...	2	2	1

5 rows × 21 columns

```
df.tail()
```

	USMER	MEDICAL_UNIT	SEX	PATIENT_TYPE	DEATH	INTUBED	PNEUMONIA	AGE	PREGNANT	DIABETES	...	ASTHMA	INMSUPR	HIPERTENSI
1048570	2	13	2	1	9999-99-99	97	2	40	97	2	...	2	2	
1048571	1	13	2	2	9999-99-99	2	2	51	97	2	...	2	2	
1048572	2	13	2	1	9999-99-99	97	2	55	97	2	...	2	2	
1048573	2	13	2	1	9999-99-99	97	2	28	97	2	...	2	2	
1048574	2	13	2	1	9999-99-99	97	2	52	97	2	...	2	2	

Jak widać po pierwszych pięciu i ostatnich wierszach baza danych jest niezrozumiała. Dobrym przykładem jest ciąża, jeśli dana osoba jest mężczyzną to wtedy kolumna "PREGNANT" zawiera liczbe 97 lub 1, a kiedy jest kobietą zawiera liczbę 2 lub 98 w zależności od tego czy jest w ciąży.

- Zmiana nazw kolumn or usunięcie niepotrzebnych. Niektóre wartości zostały zmienione na wartości binarne. Usunięte zostały także wiersze, które wskazywały że pacjent nie chorował na COVID.

```
df.rename(columns= {'DATE_DIED':"DEATH"},inplace=True)


cols =['USMER','MEDICAL_UNIT','SEX','PATIENT_TYPE','DEATH','INTUBED','
          'COPD','ASTHMA','HIPERTENSION','OTHER_DISEASE',
          'CARDIOVASCULAR','OBESITY','RENAL_CHRONIC','TOBACCO']


def change(column,points,names=None):
    if not names:
        names= range(len(points)+1)
    colCut= pd.cut(column,bins = [column.min()]+ points+[column.max()]
    return colCut

df['INTUBED']=change(df['INTUBED'],[90],[1,0])
df['PREGNANT']=change(df['PREGNANT'],[97],[0,1])
df['HIPERTENSION']=change(df['HIPERTENSION'],[90],[0,1])
df['PNEUMONIA']=change(df['PNEUMONIA'],[90],[0,1])
df['TOBACCO']=change(df['TOBACCO'],[90],[0,1])
df['OTHER_DISEASE']=change(df['OTHER_DISEASE'],[90],[0,1])
df['CARDIOVASCULAR']=change(df['CARDIOVASCULAR'],[90],[0,1])
df['OBESITY']=change(df['OBESITY'],[90],[0,1])
df['RENAL_CHRONIC']=change(df['RENAL_CHRONIC'],[90],[0,1])
df['ASTHMA']=change(df['ASTHMA'],[90],[0,1])
df['COPD']=change(df['COPD'],[90],[0,1])
df['DIABETES']=change(df['DIABETES'],[90],[0,1])
df.head()


df.drop(df.loc[df['CLASIFFICATION_FINAL']==4].index, inplace=True)
df.drop(df.loc[df['CLASIFFICATION_FINAL']==5].index, inplace=True)
df.drop(df.loc[df['CLASIFFICATION_FINAL']==6].index, inplace=True)
df.drop(df.loc[df['CLASIFFICATION_FINAL']==7].index, inplace=True)
df = df.drop('INMSUPR', axis=1)
df = df.drop('CLASIFFICATION_FINAL', axis=1)
df = df.drop('ICU', axis=1)
```

	USMER	MEDICAL_UNIT	SEX	PATIENT_TYPE	DEATH	INTUBED	PNEUMONIA	AGE	PREGNANT	DIABETES	COPD	ASTHMA	HIPERTENSION	OTHE
0	2	1	1	1	03/05/2020	1	0	65	0	0	0	0	0	
1	2	1	2	1	03/06/2020	1	0	72	0	0	0	0	0	
2	2	1	2	2	09/06/2020	0	0	55	0	0	0	0	0	
3	2	1	1	1	12/06/2020	1	0	53	0	0	0	0	0	
4	2	1	2	1	21/06/2020	1	0	68	0	0	0	0	0	

1.3. Statystyki

Średnia, minimalna i maksymalna wartość dla danej kolumny oraz odchylenie standardowe są podstawowymi danymi, które pozwolą nam wykryć ewentualne błędy.

```
df_for_stats = df.iloc[:1000000]
cols_for_stats=cols
cols_for_stats.remove("DEATH")
for col in cols:
    print("For ",col,
          ":\nMean: ",df_for_stats[col].astype('int').mean(),
         "\nMin: ",df_for_stats[col].astype('int').min(),
         "\nMax: ",df_for_stats[col].astype('int').max(),
         "\nStd: ",df_for_stats[col].astype('int').std())
```

Można zauważyć, że nie ma reguły dla osób chorujących na COVIDA. Dane są wprost proporcjonalne dla meżczyzn jak i dla kobiet.

```
For  USMER :
Mean:  1.6246584638462775
Min:  1
Max:  2
Std:  0.48421159170137884
For  MEDICAL_UNIT :
Mean:  8.697948104362734
Min:  1
Max:  13
Std:  3.764234679730042
For  SEX :
Mean:  1.5344393449649087
Min:  1
Max:  2
Std:  0.4988131576887579
For  PATIENT_TYPE :
Mean:  1.2839233734460265
Min:  1
Max:  2
Std:  0.45090066548675384
For  INTUBED :
Mean:  0.280063472788083
Min:  0
Max:  1
Std:  0.4490305539534914
For  PNEUMONIA :
Mean:  1.0204628309169623e-05
Min:  0
Max:  1
Std:  0.0031944561678228078
For  AGE :
Mean:  45.18718604823218
Min:  0
Max:  120
Std:  16.460983539751734
For  PREGNANT :
Mean:  0.0035282502378953975
Min:  0
Max:  1
Std:  0.05929427170932646
For  DIABETES :
Mean:  0.0036736661913010647
Min:  0
Max:  1
Std:  0.06049941905265965
For  COPD :
Mean:  0.003349669242484929
Min:  0
Max:  1
Std:  0.057779386249598524
For  ASTHMA :
Mean:  0.0033394646141757596
Min:  0
Max:  1
Std:  0.057691603213434015
For  HIPERTENSION :
Mean:  0.0035410060232818597
Min:  0
Max:  1
```

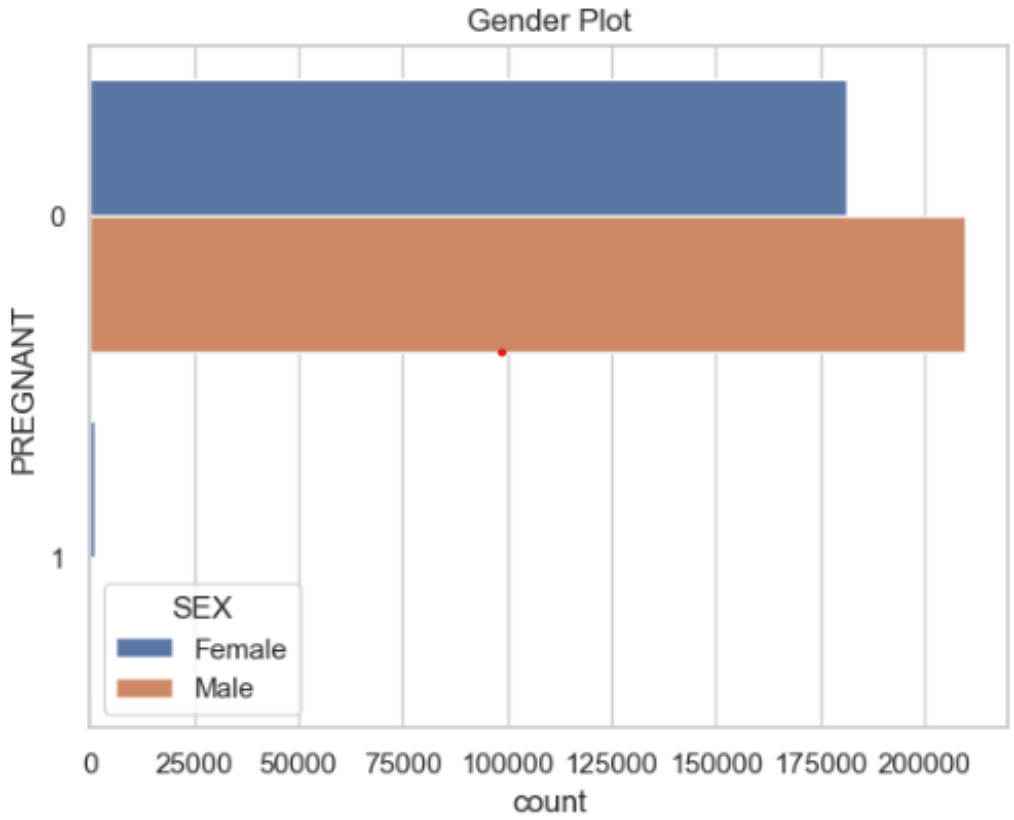
1.4. Dane na wykresach

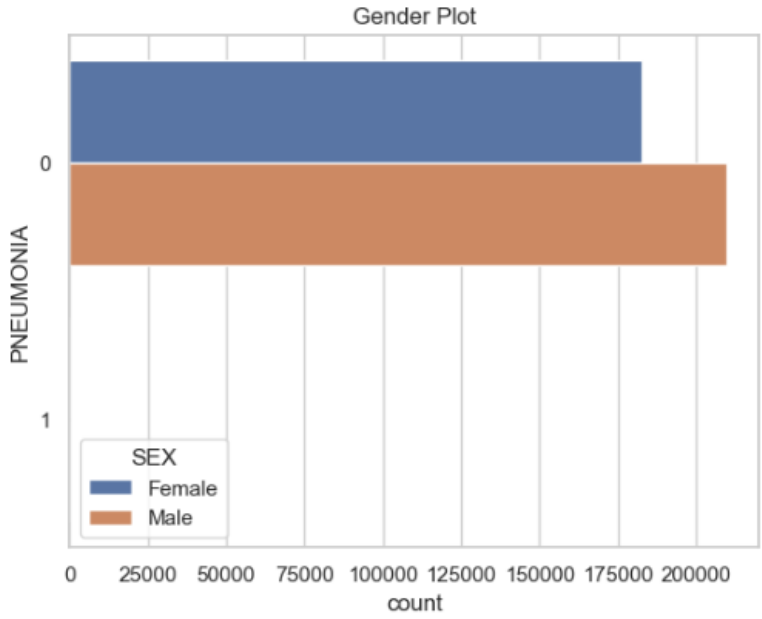
Aby baza danych była bardziej czytelna dla użytkownika została lekko zmodyfikowana.

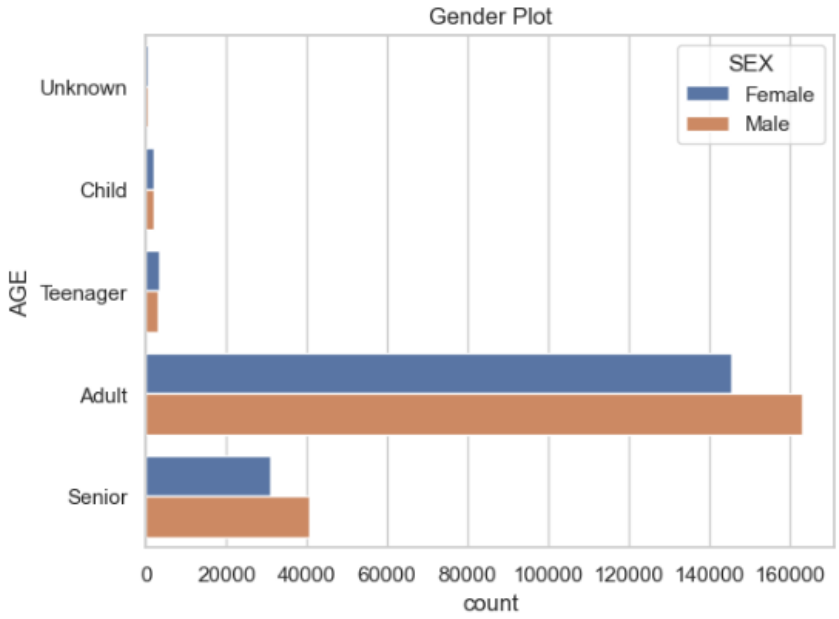
```
repSex = {1: "Female", 2: "Male"}
df.replace({"SEX": repSex},inplace=True)
df['AGE']=change(df['AGE'],[1,11,18,60],["Unknown","Child","Teenager","Adu
repDate={"9999-99-99":0}
df.replace({"DEATH":repDate},inplace=True)
df.loc[df["DEATH"] != 0,"DEATH"]=1
```

Po tych zmianach tworzymy wykresy porównawcze.

```
new_cols=cols
new_cols.remove("SEX")
for x in new_cols:
    sns.set(style="whitegrid")
    ax = sns.countplot(y=x, hue="SEX", data=df)
    plt.ylabel(x)
    plt.title('Gender Plot')
    plt.show()
```




Gender Plot
PNEUMONIA
0
1
SEX
Female
Male
0
25000
50000
75000
100000
125000
150000
175000
200000
count


Gender Plot
AGE
Unknown
Child
Teenager
Adult
Senior
SEX
Female
Male
0
20000
40000
60000
80000
100000
120000
140000
160000
count

2. Naive-Bayes

2.1. Definicja

2.1.1. Czym jest?

Naiwny Bayes jest to klasyfikator probabilistyczny, który jest oparty na założeniu o wzajemnej niezależności predykatów. Polega na "uczeniu się" w trybie uczenia z nadzorem.

Wyróżniamy trzy klasyfikatory w bibliotece scikit-learn:

- Gaussian - dla danych ciągłych
- Multinomial - dla danych dyskretnych
- Bernoulli - dla danych binarnych

Model Bayesa używa metody maksymalnego prawdopodobieństwa.

2.1.2. Wzór

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

- P(A|B) - prawdopodobieństwo, że A prawdziwe jeśli widzimy dowody na B
- P(B) - prawdopodobieństwo, że B prawdziwe jeśli widzimy dowody na A
- P(B) - prawdopodobieństwo, że B prawdziwe
- P(A) - prawdopodobieństwo, że A prawdziwe

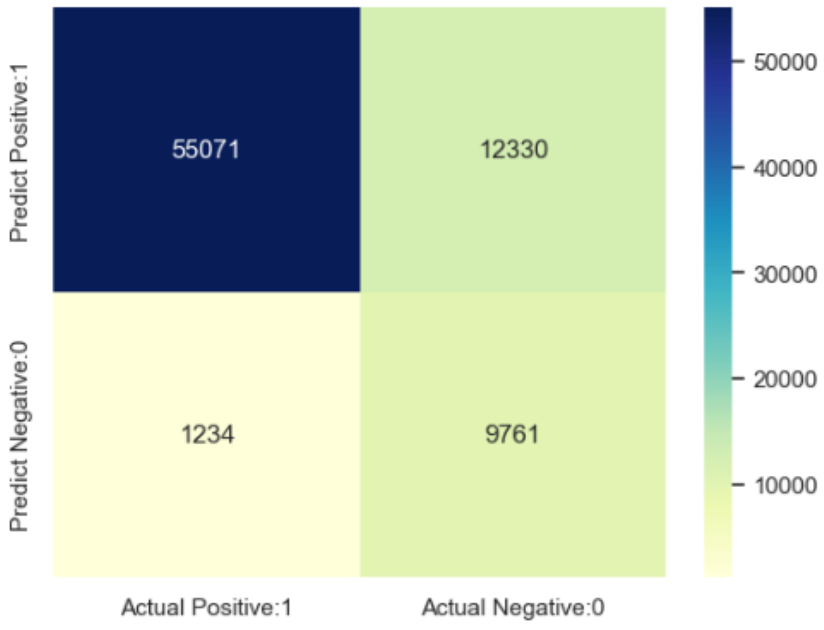
2.2. Kod

```
def naive_Bayes(X,y,typ):
    y.astype('int')
    X_train, X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,ra
    model=typ
    clf=model.fit(X_train,y_train.astype('int'))
    pred_labels=model.predict(X_test)
    print("Classes: ",clf.classes_)
    print("\n*-------------------------------------------------*\n")
    if str(typ)=='GaussianNB()':
        print("Class Priors: ", clf.class_prior_)
    else:
        print("Class Priors: ", clf.class_log_prior_)
    score=model.score(X_test,y_test.astype('int'))
    print("\n*-------------------------------------------------*\n")
    print("Score: ",score)
    print("\n*-------------------------------------------------*\n")
    print('Training set score: {:.4f}'.format(model.score(X_train, y_train
    print('Test set score: {:.4f}'.format(model.score(X_test, y_test.astyp
    print("\n*-------------------------------------------------*\n")
    print( classification_report(y_test.astype('int'),pred_labels))
    print("\n*-------------------------------------------------*\n")
    y_pred = clf.predict(X_test)
    cm = confusion_matrix(y_test.astype('int'), y_pred.astype('int'))
    cm_matrix = pd.DataFrame(data=cm, columns=['Actual Positive:1', 'Actua
                                index=['Predict Positive:1', 'Predict Neg
    sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='YlGnBu')
    return X_train,X_test,y_train.astype('int'),y_test.astype('int'),clf,p
```

- Gaussian

```
X=df['INTUBED'].values.reshape(-1,1)
y=df["DEATH"].values
X_train,X_test,y_train,y_test,clf,pred_labels,=naive_Bayes(X,y,Gaussia
```

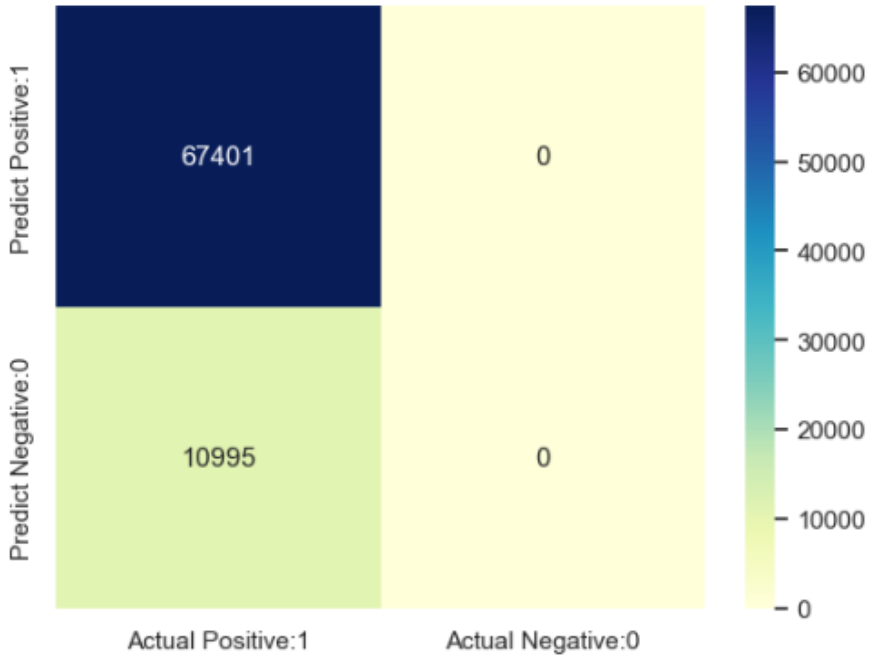
```
Classes:  [0 1]

*-------------------------------------------------*

Class Priors:  [0.86210668 0.13789332]

*-------------------------------------------------*

Score:  0.826980968416756

*-------------------------------------------------*

Training set score: 0.8262
Test set score: 0.8270

*-------------------------------------------------*

              precision    recall  f1-score   support

           0       0.98      0.82      0.89     67401
           1       0.44      0.89      0.59     10995

    accuracy                           0.83     78396
   macro avg       0.71      0.85      0.74     78396
weighted avg       0.90      0.83      0.85     78396


*-------------------------------------------------*
```



- Bernoulli

```
X=df['INTUBED'].values.reshape(-1,1)
y=df["DEATH"].values
X_train,X_test,y_train,y_test,clf,pred_labels,=naive_Bayes(X,y,Bernoul
```

```
Classes:  [0 1]

*--------------------------------------------------*

Class Priors:  [-0.14837625 -1.98127496]

*--------------------------------------------------*

Score:  0.8597504974743609

*--------------------------------------------------*

Training set score: 0.8621
Test set score: 0.8598

*--------------------------------------------------*

              precision    recall  f1-score   support

           0       0.86      1.00      0.92     67401
           1       0.00      0.00      0.00     10995

    accuracy                           0.86     78396
   macro avg       0.43      0.50      0.46     78396
weighted avg       0.74      0.86      0.79     78396


*--------------------------------------------------*
```



2.3. Wyjaśnienie

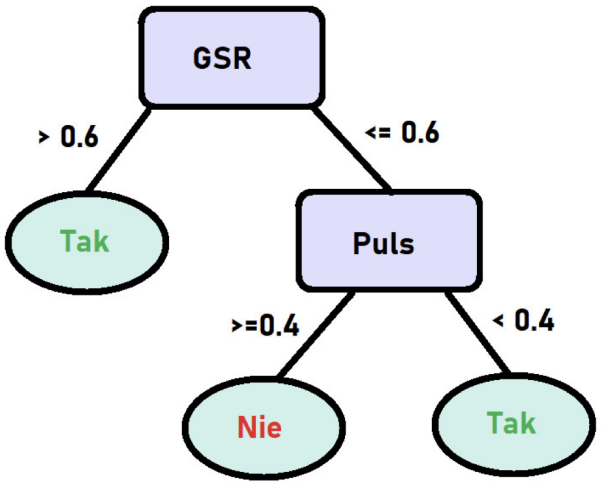
Jak można zauważyć nawiny bayes nie sprawdził się z klasyfikatorem Bernoulli. Gaussian za to wypadł o wiele lepiej. Pokazał, że jest pewna zależność między śmiercią z powodu covida, a byciem zaintubowanym. Ze względu na o wiele mniejszą ilość osób, które umarły klasyfikacja nie jest najlepsza.

3. KNN

3.1. Definicja

Metoda K najbliższych sąsiadów należy do grupy algorytmów leniwych. Polega na podporządkowaniu danej obserwacji taką klasę, która ma najwięcej podobnych próbek.

Ciekawym przykładem może być klasyfikacja czy dany człowiek skłamał poprzez ewaulacje pulsu wraz z badaniami galwanometrem.



Zbiór treningowy

Puls	GSR	Winny
1	0,7	Tak
0,8	0,8	Tak
0,9	0,9	Tak
0,6	1	Tak
0,5	0,5	Tak
0,3	0,9	Tak
0,3	0,4	Nie
0,2	0	Nie
0,1	0,2	Nie
0	0,3	Nie
0,6	0,8	Nie

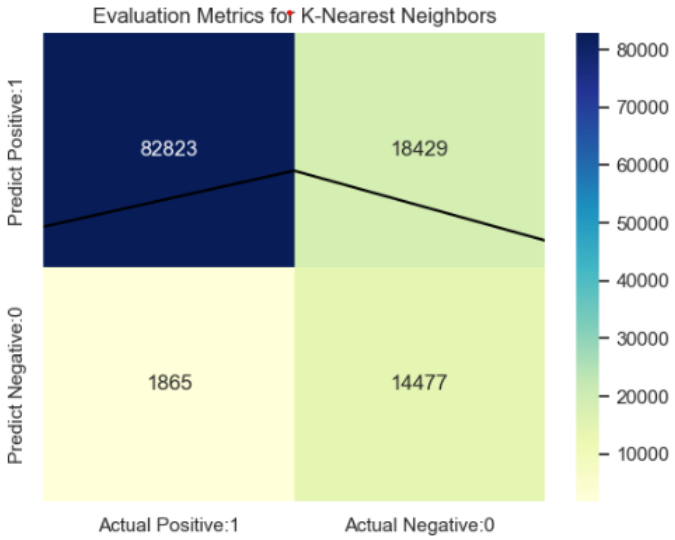
Zbiór testowy

Puls	GSR	Winny
0,4	0,6	Nie
0,6	0,6	Tak
0,4	0,9	Tak
0,5	0,2	Nie
0,5	0,6	Tak

3.2. Kod

```
def knn(X,Y):
    X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.3
    knn_model = KNeighborsClassifier(n_neighbors=20)
    knn_model.fit(X_train, Y_train.astype("int"))
    Y_predict_knn = knn_model.predict(X_test)
    #Comparing the output I expected (Y_test) against the ones the model p
    knn_metrics = metrics.classification_report(Y_test.astype("int"),Y_pre
    print(knn_metrics)
    table = pd.DataFrame(Y_test.astype("int"))
    print('table 1')
    print(table.head())
    #add the predictions to the dataframe
    table['predictions'] = Y_predict_knn.astype("int")
    print('table 2')
    print(table.head())
    accuracy_knn = accuracy_score(Y_test.astype("int"),Y_predict_knn.astyp
    precision_knn = precision_score(Y_test.astype("int"), Y_predict_knn.as
    f1_knn = f1_score(Y_test.astype("int"),Y_predict_knn.astype("int"))
    recall_knn = recall_score(Y_test.astype("int"), Y_predict_knn.astype('
    print(precision_knn)
    print(accuracy_knn)
    print(f1_knn)
    print(recall_knn)
    plt.bar(['Accuracy','F1 Score','Recall Score','Precision Score'],[accu
    plt.plot([accuracy_knn,f1_knn,recall_knn,precision_knn],color='black')
    plt.title('Evaluation Metrics for K-Nearest Neighbors')
    plt.show
    cm = confusion_matrix(Y_test.astype('int'), Y_predict_knn.astype('int
    cm_matrix = pd.DataFrame(data=cm, columns=['Actual Positive:1', 'Actua
                                index=['Predict Positive:1', 'Predict Neg
    sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='YlGnBu')

knn(X=df["INTUBED"].iloc[:100000].values.reshape(-1,1),
Y = df["DEATH"].iloc[:100000].values)
```

```
              precision    recall  f1-score   support

           0       0.98      0.82      0.89    101252
           1       0.44      0.89      0.59     16342

    accuracy                           0.83    117594
   macro avg       0.71      0.85      0.74    117594
weighted avg       0.90      0.83      0.85    117594

table 1
   0
0  0
1  0
2  0
3  0
4  0
table 2
   0  predictions
0  0            0
1  0            0
2  0            0
3  0            0
4  0            0
Precission:  0.4399501610648514
Accuracy:  0.8274231678486997
F1:  0.5879223521767382
Recall:  0.8858768816546322
```



3.3. Wyjaśnienia

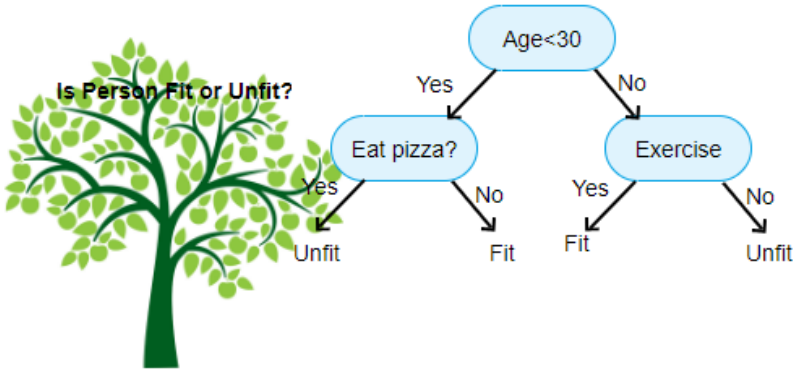
Powyższy algorytm KNN zawiera 20 sąsiadów. Wypadł bardzo podobnie do algorytmu naiwnego-bayesa. Objaśnienia wyników:

- Accuracy - odsetek trafionych predykcji
- Precision - stosunek tp / (tp + fp), gdzie tp to liczba prawdziwych trafień, a fp liczba fałszywych trafień.
- Recall - stosunek tp / (tp + fn) gdzie tp to liczba wyników prawdziwie dodatnich, a fn liczba wyników fałszywie ujemnych.
- F1 Score - 2((precision*recall)/(precision+recall))

4. Decision-Tree

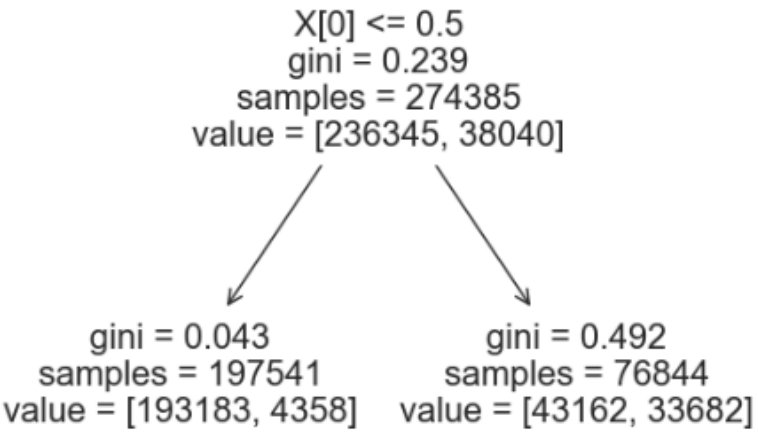
4.1. Definicja

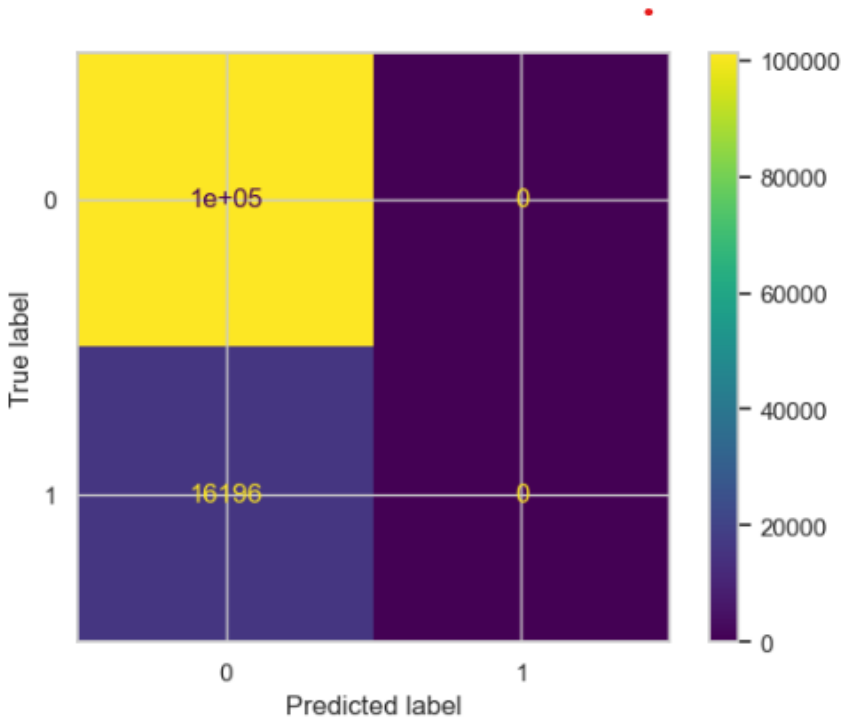
Drzewo decyzyjne jest to jeden ze sposobów klasyfikacji, polegający na podejmowaniu decyzji na podstawie pytań. Przykładem może byc klasyfikacja czy człowiek prowadzi zdrowy tryb życia.



4.2. Kod

```
X=df['INTUBED'].values.reshape(-1,1)
y = df["DEATH"].values.astype("int")
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
tree.plot_tree(clf)
dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True,feature_names = ["OTHER_DISEASE"]
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('covid_DT1.png')
Image(graph.create_png())
print("Score: ", clf.score(X,y))
plot_confusion_matrix(clf,X_test,y_test)
```





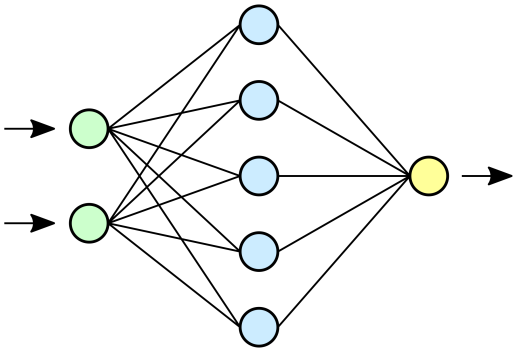
4.3. Wyjaśnienia

Drzewa decyzyjne miały wynik równy 0.861. Macierz błedu nie wygląda najlepiej, ponieważ są zera w dwóch miejscach.

5. Neural-Networks

5.1. Definicja

Sieci neuronowe wzorowane są na budowie biologicznego systemu neuronowego w ujęciu matematyczno-informatycznym są grafem skierowanym.

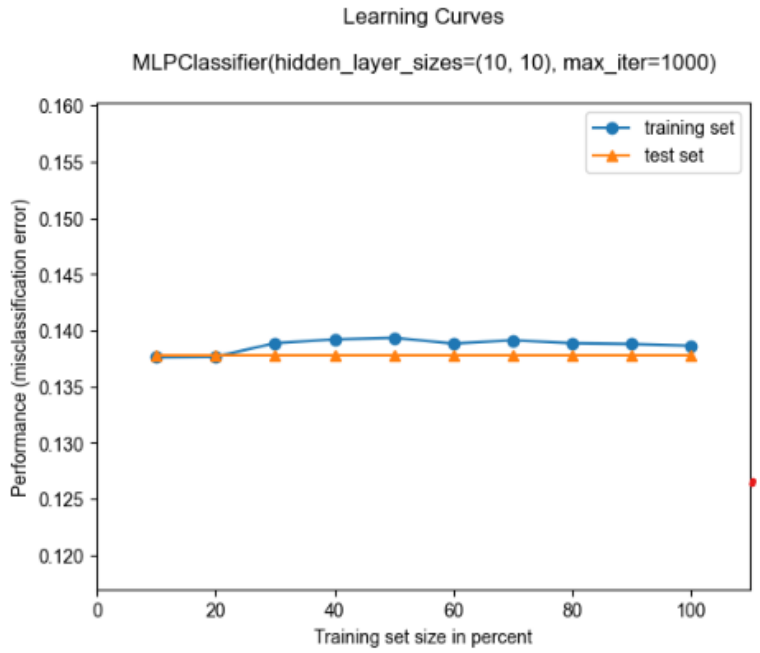


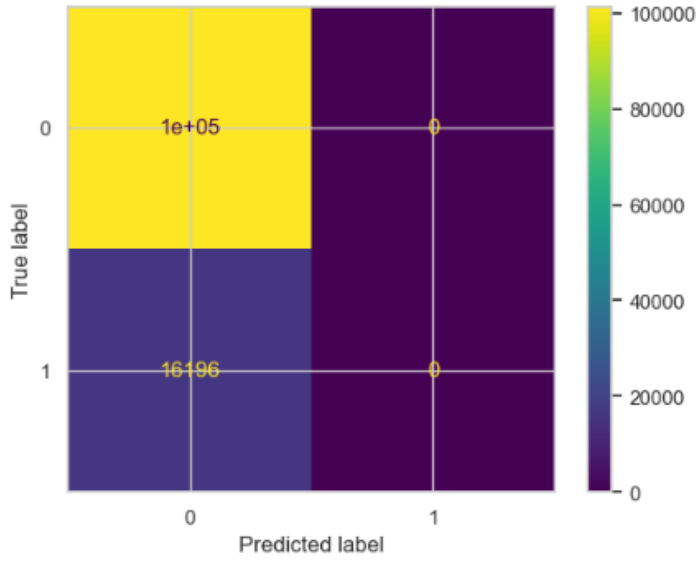
5.2. Kod

```
X=df['INTUBED'].values.reshape(-1,1)
y = df["DEATH"].values.astype("int")
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, r
def neural_network(hidden_layer_sizes,max_iter,activation,solver,learning_
    scaler = StandardScaler()

    scaler.fit(X_train)

    train_data = scaler.transform(X_train)
    test_data = scaler.transform(X_test)
    print(train_data[:3])

    mlp = MLPClassifier(hidden_layer_sizes=hidden_layer_sizes, max_iter=ma
                        activation=activation,
                        solver=solver,
                        learning_rate=learning_rate)

    mlp.fit(train_data, y_train)

    predictions_train = mlp.predict(train_data)
    predictions_test = mlp.predict(test_data)
    percent = (mlp.score(test_data, y_test))
    print("Percent: ",percent)
    plot_learning_curves(X_train, y_train, X_test, y_test,mlp)
    plt.show()
    return ["Neural Network", percent, mlp]
```

```
r=neural_network((10, 10),1000,'relu','adam','constant',X,y)
plot_confusion_matrix(r[2],X_test,y_test)
```

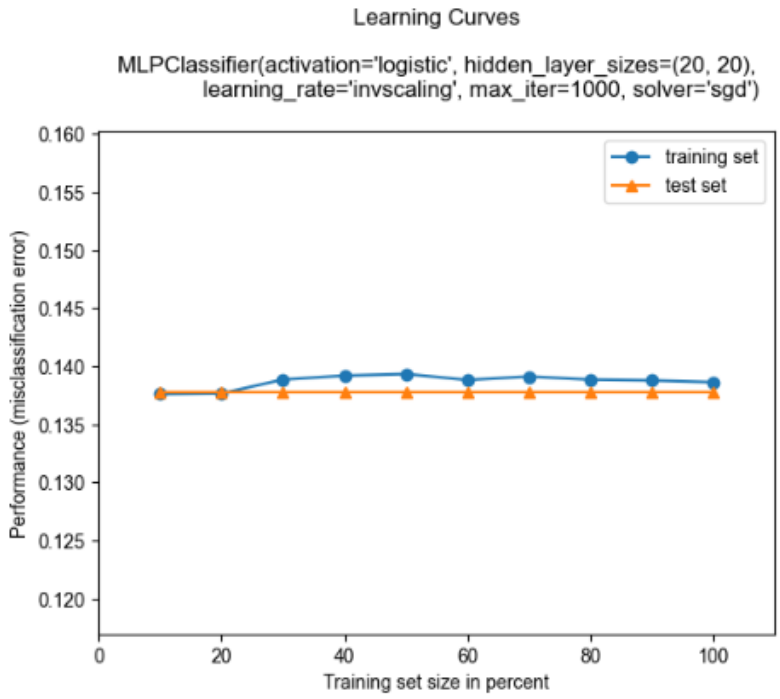
```
[[ 1.60333267]
 [-0.62370088]
 [ 1.60333267]]
Percent:  0.8622718846199636
```

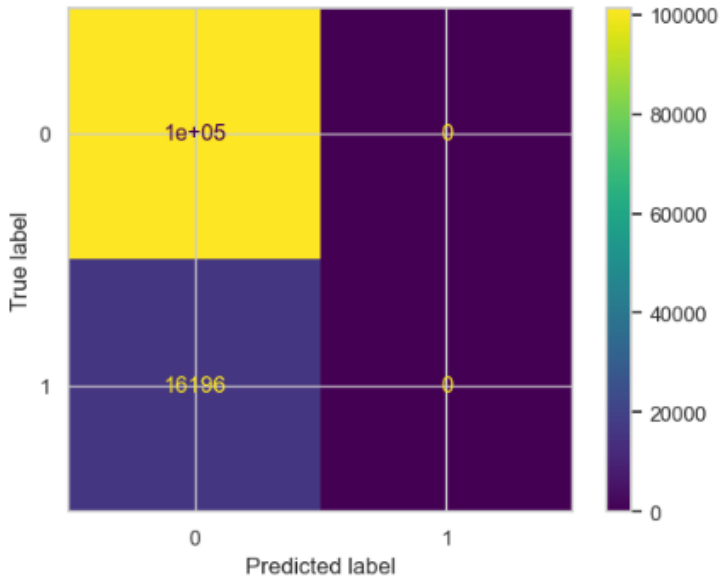




```
r=neural_network((20, 20),1000,'logistic','sgd','invscaling',X,y)
plot_confusion_matrix(r[2],X_test,y_test)
```

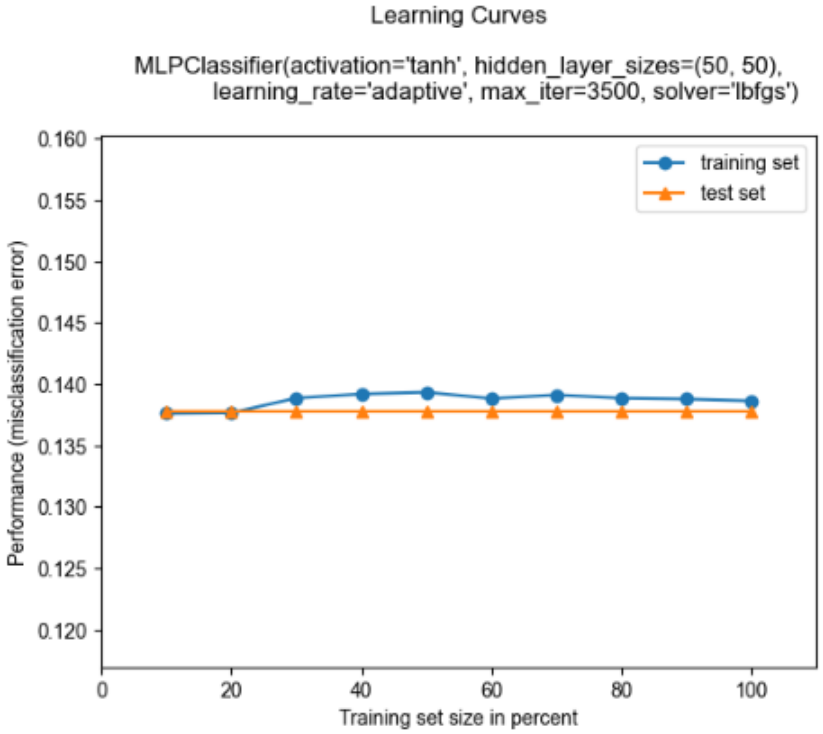
```
[[ 1.60333267]
 [-0.62370088]
 [ 1.60333267]]
Percent:  0.8622718846199636
```

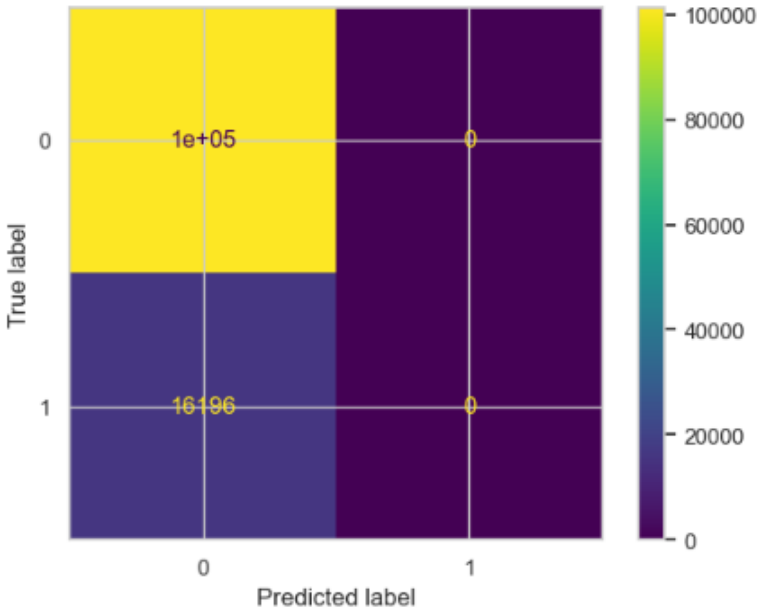




```
r=neural_network((50, 50),3500,'tanh','lbfgs','adaptive',X,y)
plot_confusion_matrix(r[2],X_test,y_test)
```

```
[[ 1.60333267]
 [-0.62370088]
 [ 1.60333267]]
Percent:  0.8622718846199636
```



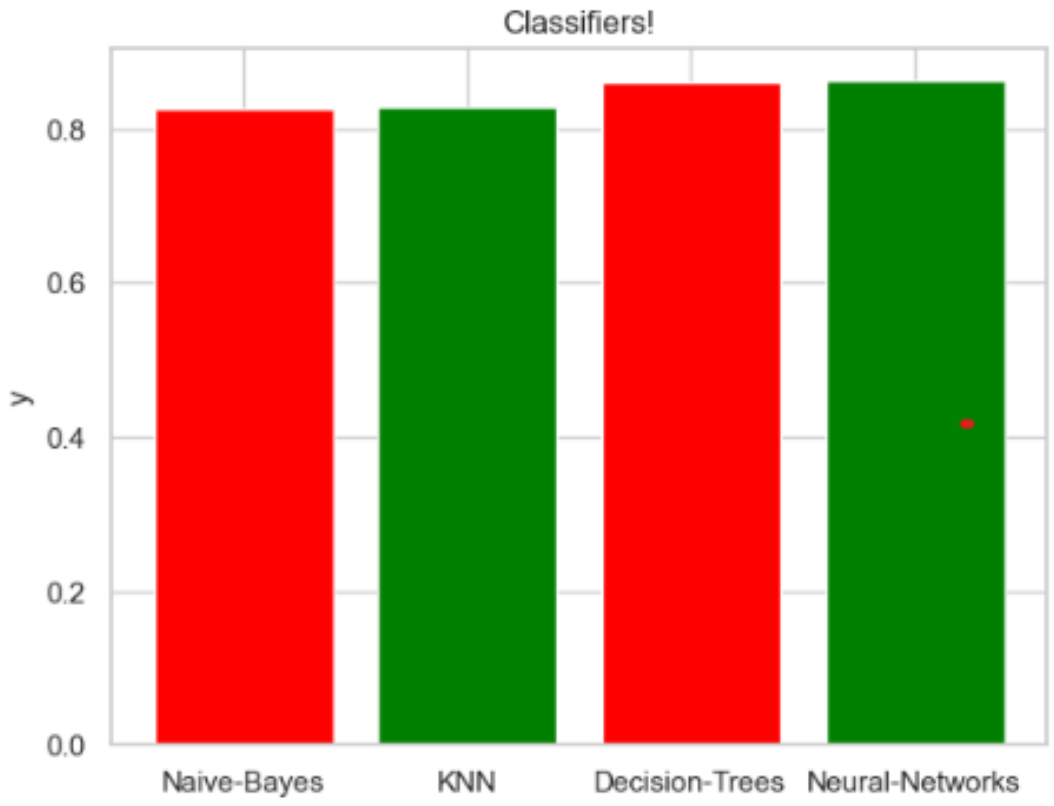


5.3. Wyjaśnienia

Macierz błedu ponownie nie wygląda najciekawiej, chociaż osiągnęła najlepszy wynik. Wyniki zostały przedstawione także na krzywej uczenia się, czyli zależności opanowania pewnej umiejętności w danym czasie. Nie ma wielkich różnic między innymi funkcjami aktywacji, róznymi wskaźnikami uczenia się itp.

6. Podsumowanie

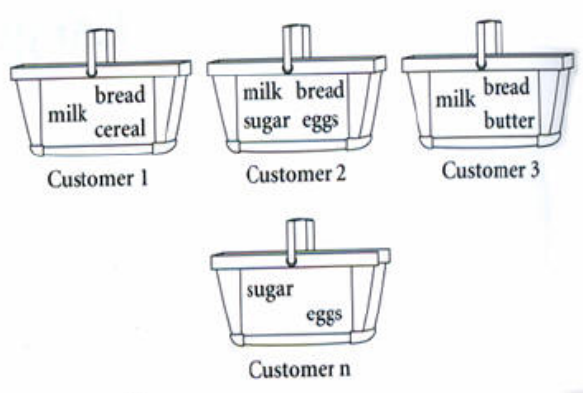
Jak widać na wykresie najlepiej sprawdziły się sieci neuronowe mimo nienajlepszej macierzy błedu. Najciekawszy dla mnie był algorytm naiwnego-bayesa, ponieważ nie bierze pod uwage żadnych zależności tylko stosuje techniki bayesowskie.



7. Apriori

7.1. Definicja

Reguły asocjacyjne polegają na ocenie wiarygodności jakiejś reguły. Najlepiej wytłumaczyć takie reguły na bazie danych:



Tid	Towary
1	Bread, Milk
2	Beer, Diaper, Bread, Eggs
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Bread, Diaper, Milk

$\Longrightarrow$

Przykłady reguł asocjacyjnych:

$$\{Coke\} \Longrightarrow \{Diaper\}$$
$$\{Diaper, Milk\} \Longrightarrow \{Coke\}$$
$$\{Milk\} \Longrightarrow \{Bread, Beer\}$$

„Kiedy kupimy pieluche i mleko, wtedy też kupimy piwo" - stwierdzenie to jest prawdziwe tylko dla 3 i 4, a 5 nie zawiera piwa więc wiarygoność jest równa 2/3.

Tid	Towary
1	Bread, Milk
2	Beer, Diaper, Bread, Eggs
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Bread, Diaper, Milk

$\Longrightarrow$

Reguła assocjacyjna: $\{Diaper, Milk\} \Longrightarrow Beer$

Wsparcie: $s(Diaper, Milk, Beer) = \frac{2}{5} = 0.4$

Wiarygodność:

$$\frac{s(Diaper, Milk, Beer)}{s(Diaper, Milk)} = \frac{2}{3} = 0.67$$

7.2. Kod

Aby rozpocząć trzeba przygotować baze danych. Wartości stają się kolumnami:

```
data = []
df_te=df.iloc[:2000]
df_te['PREGNANT']=change(df_te['PREGNANT'],[0.5],['NOT PREGNANT','PREGNANT
df_te['TOBACCO']=change(df_te['TOBACCO'],[0.5],["NOT TOBACOO","TOBACCO"])
df_te['OTHER_DISEASE']=change(df_te['OTHER_DISEASE'],[0.5],["NOT OTHER DIS
df_te['OBESITY']=change(df_te['OBESITY'],[0.5],["NOT OBESITY","OBESITY"])
df_te['ASTHMA']=change(df_te['ASTHMA'],[0.5],['NOT ASTHMA','ASTHMA'])
df_te['DIABETES']=change(df_te['DIABETES'],[0.5],["NOT DIABETES","DIABETES
df_te['DEATH']=change(df_te['DEATH'],[0.5],["NO","YES"])
df_te = df_te.drop('MEDICAL_UNIT', axis=1)
df_te = df_te.drop('INTUBED', axis=1)
df_te = df_te.drop('USMER', axis=1)
df_te = df_te.drop('CARDIOVASCULAR', axis=1)
df_te = df_te.drop('HIPERTENSION', axis=1)
df_te = df_te.drop('PNEUMONIA', axis=1)
df_te = df_te.drop('RENAL_CHRONIC', axis=1)
df_te = df_te.drop('PATIENT_TYPE', axis=1)
df_te = df_te.drop('COPD', axis=1)

for i in range(0, df_te.shape[0]-1):
    data.append([str(df_te.values[i,j]) for j in range(0, df_te.shape[1])]


th = TransactionEncoder()
th_arr = th.fit(data).transform(data)
new_df = pd.DataFrame(th_arr,columns=th.columns_)
new_df.head()
```

	ASTHMA	Adult	Child	DIABETES	Female	INTUBED	Male	NO	NOT ASTHMA	NOT DIABETES	...	I
0	False	False	False	False	True	True	False	False	True	True	...	
1	False	False	False	False	False	True	True	False	True	True	...	
2	False	True	False	False	False	False	True	False	True	True	...	
3	False	True	False	False	True	True	False	False	True	True	...	
4	False	False	False	False	False	True	True	False	True	True	...	

Wyniki aprori:

```
apr = apriori(new_df,min_support = 0.2, use_colnames = th.columns_)
apr.head()
```

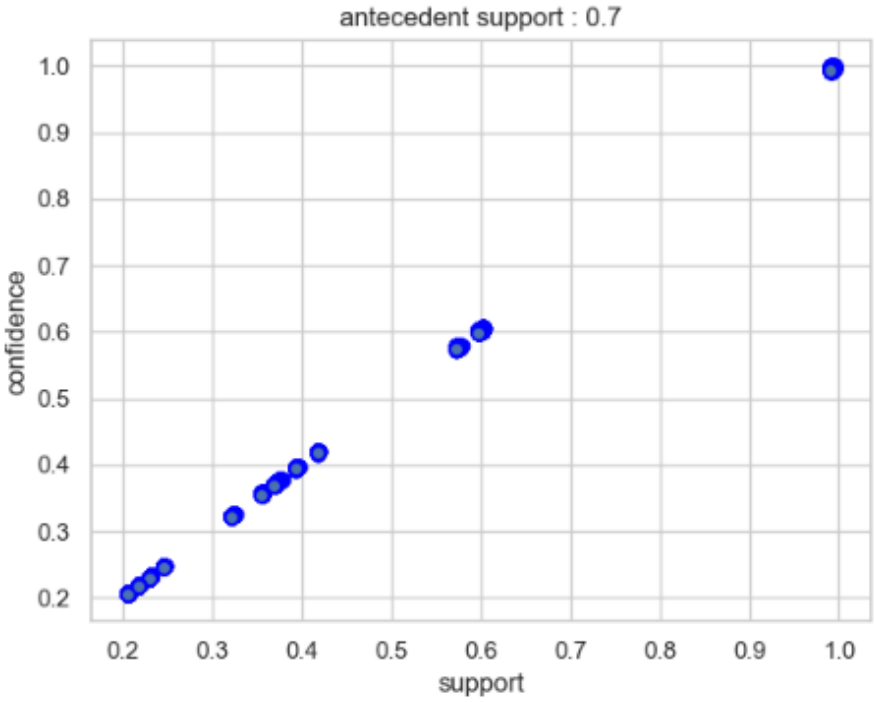
	support	itemsets
0	0.604802	(Adult)
1	0.397699	(Female)
2	0.602301	(Male)
3	0.419710	(NO)
4	0.996498	(NOT ASTHMA)

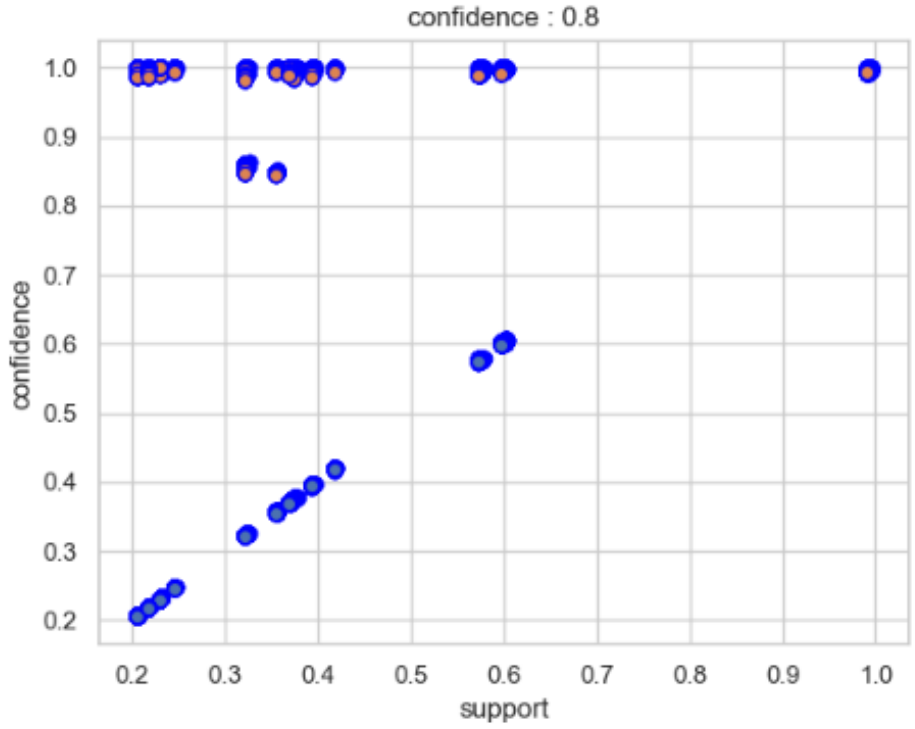
Uruchamianie eksploracji reguł z konfiguracją:

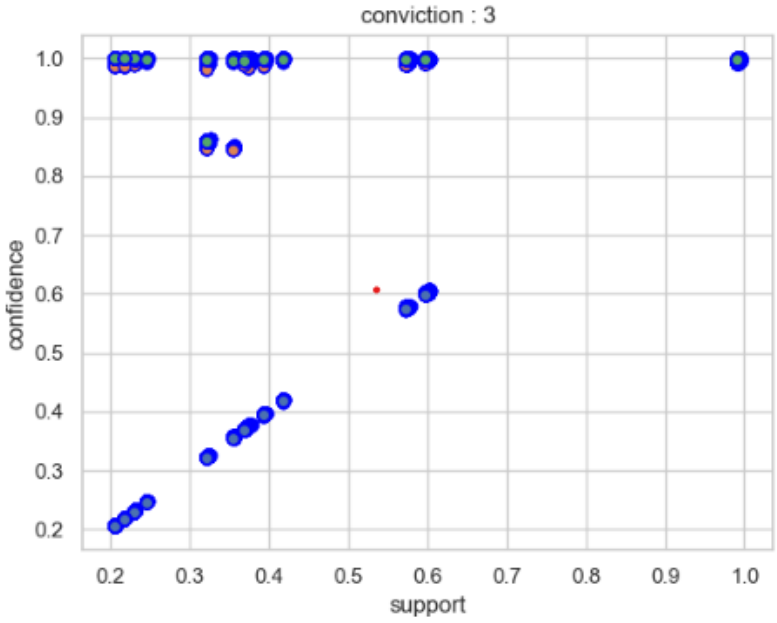
```
config = [ ('antecedent support',0.7),('confidence',0.8),('conviction',3)]
for metric, new_th in config:
    rules = association_rules(apr, metric = metric, min_threshold=new_th)
    if rules.empty:
        print("Dataframe is Empty")
    print(rules.columns.values)
    print("My configuration: ", metric, " : ",new_th)
    print(rules)

    support = rules.loc[:,"support"]
    confidence = rules.loc[:,'confidence']
    plt.scatter(support,confidence,edgecolors="blue")
    plt.xlabel('support')
    plt.ylabel('confidence')
    plt.title(metric+' : ' +str(new_th))
    plt.savefig('plot%03s.png'%(metric))
```

```
['antecedents' 'consequents' 'antecedent support' 'consequent support'
 'support' 'confidence' 'lift' 'leverage' 'conviction']
My configuration:  antecedent support  :  0.7
               antecedents                                        consequents  \
0              (NOT ASTHMA)                                           (Adult)
1            (NOT DIABETES)                                           (Adult)
2             (NOT OBESITY)                                           (Adult)
3       (NOT OTHER DISEASE)                                           (Adult)
4            (NOT PREGNANT)                                           (Adult)
...                     ...                                              ...
10572        (NOT PREGNANT)  (NOT DIABETES, YES, NOT OTHER DISEASE, NOT TOB...
10573   (NOT OTHER DISEASE)  (NOT DIABETES, NOT PREGNANT, YES, NOT TOBACOO,...
10574         (NOT TOBACOO)  (NOT DIABETES, NOT PREGNANT, YES, NOT OTHER DI...
10575          (NOT ASTHMA)  (NOT DIABETES, NOT PREGNANT, YES, NOT OTHER DI...
10576         (NOT OBESITY)  (NOT DIABETES, NOT PREGNANT, YES, NOT OTHER DI...

       antecedent support  consequent support   support  confidence      lift  \
0                0.996498            0.604802  0.603302    0.605422  1.001024
1                0.995998            0.604802  0.603302    0.605726  1.001527
2                0.996998            0.604802  0.603302    0.605118  1.000522
3                0.993997            0.604802  0.602801    0.606442  1.002711
4                0.999500            0.604802  0.604302    0.604605  0.999673
...                   ...                 ...       ...         ...       ...
10572            0.999500            0.321161  0.321161    0.321321  1.000501
10573            0.993997            0.323662  0.321161    0.323100  0.998265
10574            0.995498            0.321661  0.321161    0.322613  1.002960
10575            0.996498            0.321161  0.321161    0.322289  1.003514
10576            0.996998            0.321161  0.321161    0.322127  1.003011

       leverage  conviction
0      0.000617    1.001569
1      0.000920    1.002342
2      0.000315    1.000799
3      0.001630    1.004166
4     -0.000198    0.999500
...         ...         ...
10572  0.000161    1.000237
10573 -0.000558    0.999170
10574  0.000948    1.001406
10575  0.001125    1.001665
10576  0.000964    1.001426

[10577 rows x 9 columns]
['antecedents' 'consequents' 'antecedent support' 'consequent support'
 'support' 'confidence' 'lift' 'leverage' 'conviction']
My configuration:  confidence  :  0.8
                     antecedents  \
0                             (NO)
1                         (Adult)
2                         (Adult)
3                         (Adult)
4                         (Adult)
...                           ...
12030  (Senior, NOT OTHER DISEASE)
12031        (NOT TOBACOO, Senior)
12032         (NOT ASTHMA, Senior)
12033        (NOT OBESITY, Senior)
12034                     (Senior)

                                    consequents  antecedent support  \
0                                       (Adult)            0.419710
1                                    (NOT ASTHMA)            0.604802
2                                  (NOT DIABETES)            0.604802
3                                   (NOT OBESITY)            0.604802
4                              (NOT OTHER DISEASE)            0.604802
```

```
       consequent support   support  confidence      lift  leverage  \
0                0.604802  0.357179    0.851013  1.407093  0.103337
1                0.996498  0.603302    0.997519  1.001024  0.000617
2                0.995998  0.603302    0.997519  1.001527  0.000920
3                0.996998  0.603302    0.997519  1.000522  0.000315
4                0.993997  0.602801    0.996691  1.002711  0.001630
...                   ...       ...         ...       ...       ...
12030            0.575788  0.321161    0.857143  1.488643  0.105420
12031            0.574287  0.321161    0.853723  1.486579  0.105121
12032            0.573287  0.321161    0.852590  1.487196  0.105210
12033            0.573287  0.321161    0.851459  1.485224  0.104923
12034            0.573287  0.321161    0.848085  1.479338  0.104063

       conviction
0        2.652566
1        1.411206
2        1.612806
3        1.209605
4        1.814407
...           ...
12030    2.969485
12031    2.910328
12032    2.894731
12033    2.872695
12034    2.808887

[12035 rows x 9 columns]
['antecedents' 'consequents' 'antecedent support' 'consequent support'
 'support' 'confidence' 'lift' 'leverage' 'conviction']
My configuration:  conviction  :  3
                                          antecedents  \
0                                              (Male)
1                                                (NO)
2                                       (NOT DIABETES)
3                                         (NOT ASTHMA)
4                                        (NOT OBESITY)
...                                                ...
4995      (NOT TOBACOO, Senior, YES, NOT OTHER DISEASE)
4996  (NOT TOBACOO, NOT ASTHMA, Senior, NOT OTHER DI...
4997  (NOT TOBACOO, NOT OBESITY, Senior, NOT OTHER D...
4998          (NOT DIABETES, Senior, NOT OTHER DISEASE)
4999           (NOT TOBACOO, Senior, NOT OTHER DISEASE)

                                          consequents  antecedent support  \
0                                       (NOT PREGNANT)            0.602301
1                                        (NOT TOBACOO)            0.419710
2                                         (NOT ASTHMA)            0.995998
3                                       (NOT DIABETES)            0.996498
4                                         (NOT ASTHMA)            0.996998
...                                                ...                 ...
4995  (NOT DIABETES, NOT ASTHMA, NOT OBESITY, NOT PR...            0.321661
4996     (NOT DIABETES, NOT OBESITY, YES, NOT PREGNANT)            0.373687
4997      (NOT DIABETES, NOT ASTHMA, YES, NOT PREGNANT)            0.373687
4998  (NOT PREGNANT, YES, NOT TOBACOO, NOT ASTHMA, N...            0.373687
4999  (NOT DIABETES, NOT PREGNANT, YES, NOT ASTHMA, ...            0.373687

      consequent support   support  confidence      lift  leverage  conviction
0                0.999500  0.602301    1.000000  1.000501  0.000301         inf
1                0.995498  0.419210    0.998808  1.003325  0.001389    3.777389
2                0.996498  0.995498    0.999498  1.003010  0.002987    6.971986
3                0.995998  0.995498    0.998996  1.003010  0.002987    3.985993
4                0.996498  0.995998    0.998996  1.002507  0.002491    3.489495
...                   ...       ...         ...       ...       ...         ...
4995             0.994497  0.321161    0.998445  1.003969  0.001270    3.538269
4996             0.577289  0.321161    0.859438  1.488749  0.105435    3.007289
4997             0.576788  0.321161    0.859438  1.490040  0.105622    3.010848
4998             0.576288  0.321161    0.859438  1.491333  0.105809    3.014407
4999             0.576788  0.321161    0.859438  1.490040  0.105622    3.010848
```


antecedent support : 0.7
confidence
support
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0


confidence : 0.8
confidence
support
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0



Reguły:

```
print(rules[rules['antecedents']==frozenset({'Male'})].to_string())
print("\n------------------------------------------\n")
print(rules[rules['antecedents']==frozenset({'Adult'})].to_string())
print("\n------------------------------------------\n")
```

```
                          antecedents                 consequents  antecedent support  consequent support   support  confidence      lift  leverage
conviction
489   (Male, Senior, NOT OTHER DISEASE)               (NOT OBESITY)            0.218109            0.996998  0.218109         1.0  1.003011  0.000655
inf
501   (Male, Senior, NOT OTHER DISEASE)              (NOT PREGNANT)            0.218109            0.999500  0.218109         1.0  1.000501  0.000109
inf
1650  (Male, Senior, NOT OTHER DISEASE)  (NOT OBESITY, NOT PREGNANT)            0.218109            0.996498  0.218109         1.0  1.003514  0.000764
inf

------------------------------------------

  antecedents     consequents  antecedent support  consequent support   support  confidence      lift  leverage  conviction
0      (Male)  (NOT PREGNANT)            0.602301              0.9995  0.602301         1.0  1.000501  0.000301         inf
```

8. Link do githuba

Cały projekt będący elementem mojej pracy został wstawiony na githuba https://github.com/komolcia/INF-D-2023-Julia-Komorowska-266386.

9. Bibliografia

- https://www.kaggle.com/code/bhanuchanderu/data-mining-a-demo-with-titanic-data/notebook - Apriori
- https://towardsdatascience.com/naive-bayes-classifier-how-to-successfully-use-it-in- - Naive Bayes
- https://pl.wikipedia.org/wiki/Naiwny_klasyfikator_bayesowski - definicja Naiwnego Bayesa
- https://www.kaggle.com/code/prashant111/knn-classifier-tutorial -KNN
- https://scikit-learn.org/stable/modules/neural_networks_supervised.html- Neural Networks
- https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html - Dokumentacja sieci neuronowych
- #https://www.projectpro.io/recipes/plot-learning-curve-in-python - krzywa uczenia się