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** 4
- 1.1 Importowanie paczek . . . 4
- 1.2 Preprocessing . . . 5
- 1.3 Statystyki . . . 7
- 1.4 Dane na wykresach . . . 9

2 **Naive-Bayes** 11
- 2.1 Definicja . . . 11
  - 2.1.1 Czym jest? . . . 11
  - 2.1.2 Wzór . . . 11
- 2.2 Kod . . . 12

3 **KNN** 15
- 3.1 Definicja . . . 15
- 3.2 Kod . . . 16

4 **Decision-Tree** 18
- 4.1 Definicja . . . 18
- 4.2 Kod . . . 18

5 **Neural-Networks** 19
- 5.1 Definicja . . . 19
- 5.2 Kod . . . 20

6 **Apriori** 22
- 6.1 Definicja . . . 22
- 6.2 Kod . . . 23

7 **Link do githuba** 29

8 **Bibliografia** 29

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
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
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
```

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

```
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	

- Zmiana nazw kolumn or usunięcie niepotrzebnych. Niektóre wartości zostały zmienione na wartości binarne.

```
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],[0,1])
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 = 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())
```

```
For  SEX :
Mean:  1.501222
Min:  1
Max:  2
Std:  0.49999875671321103
For  PATIENT_TYPE :
Mean:  1.197682
Min:  1
Max:  2
Std:  0.39825115879302275
For  INTUBED :
Mean:  0.809557
Min:  0
Max:  1
Std:  0.39265075821347645
For  PNEUMONIA :
Mean:  0.015837
Min:  0
Max:  1
Std:  0.12484472362581059
For  AGE :
Mean:  41.929601
Min:  0
Max:  121
Std:  16.941643603856352
For  PREGNANT :
Mean:  0.003565
Min:  0
Max:  1
Std:  0.05960112689617811
For  DIABETES :
Mean:  0.003266
Min:  0
Max:  1
Std:  0.05705555625297591
For  COPD :
Mean:  0.002947
Min:  0
Max:  1
Std:  0.0542062554445345
For  ASTHMA :
Mean:  0.002924
Min:  0
Max:  1
Std:  0.053994936238995025
For  HIPERTENSION :
Mean:  0.003045
Min:  0
Max:  1
Std:  0.05509746827877857
For  OTHER_DISEASE :
Mean:  0.00493
Min:  0
Max:  1
Std:  0.07004070249290768
```

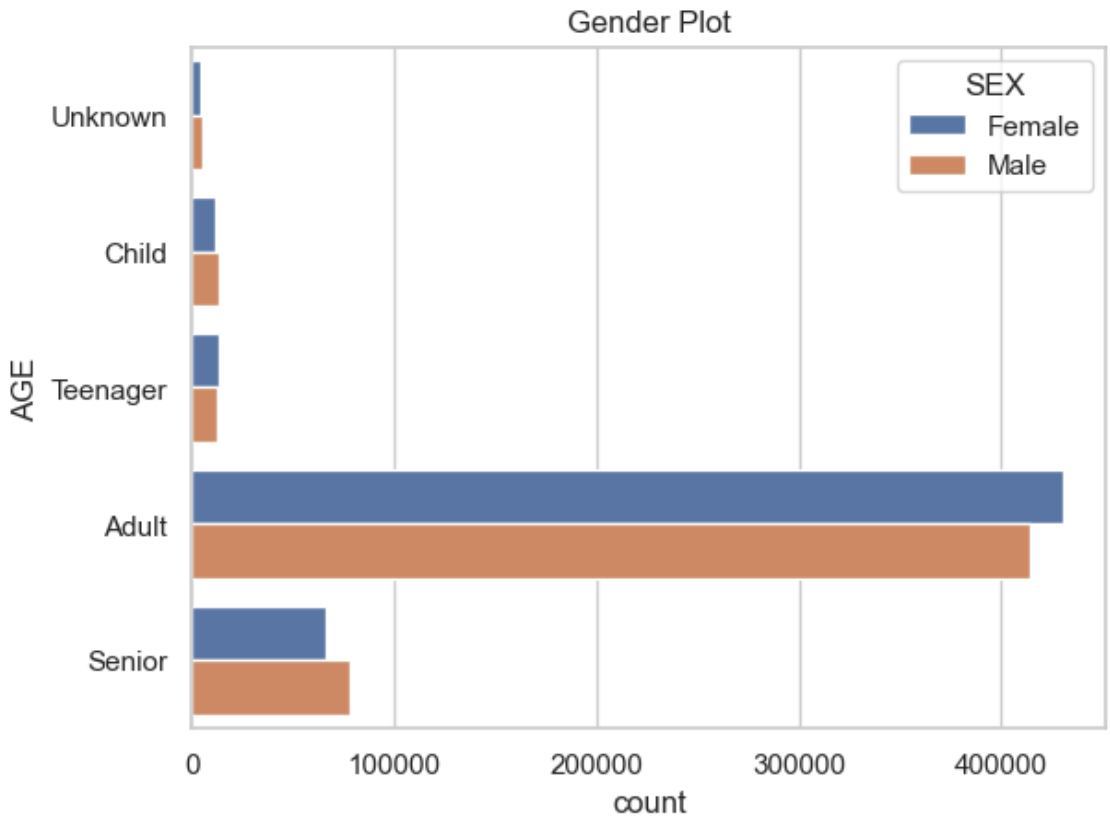
1.4. Dane na wykresach

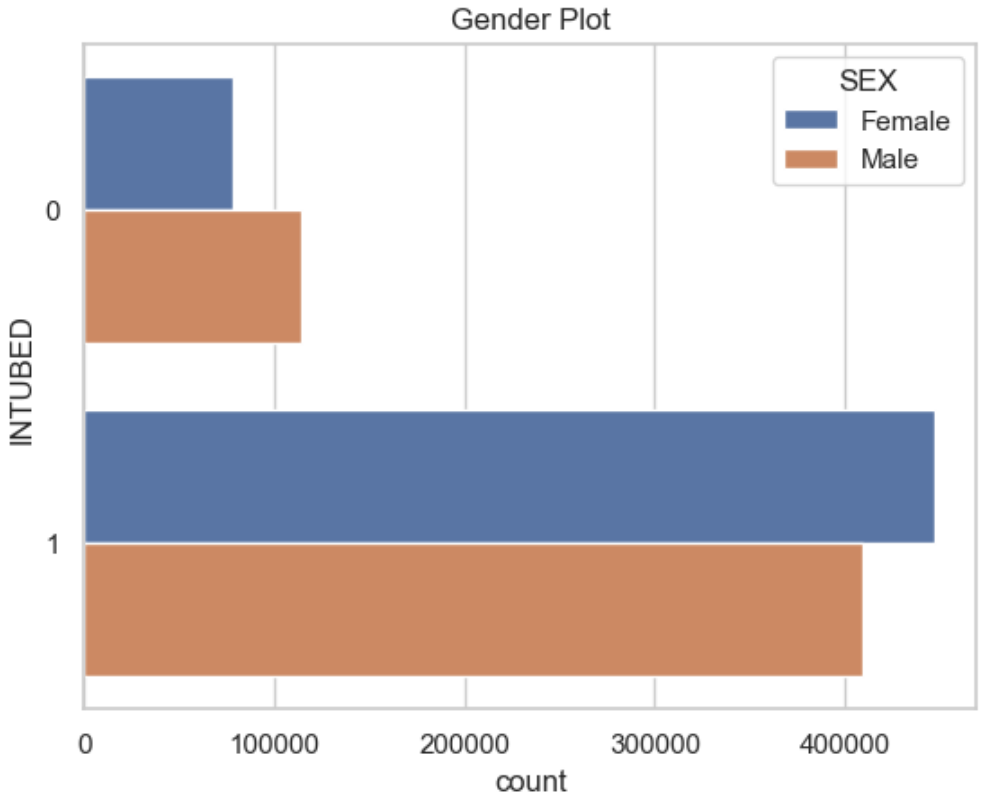
Aby baza danych była bardziej czytelna dla użytkownika została lekko zmodyfikowana.

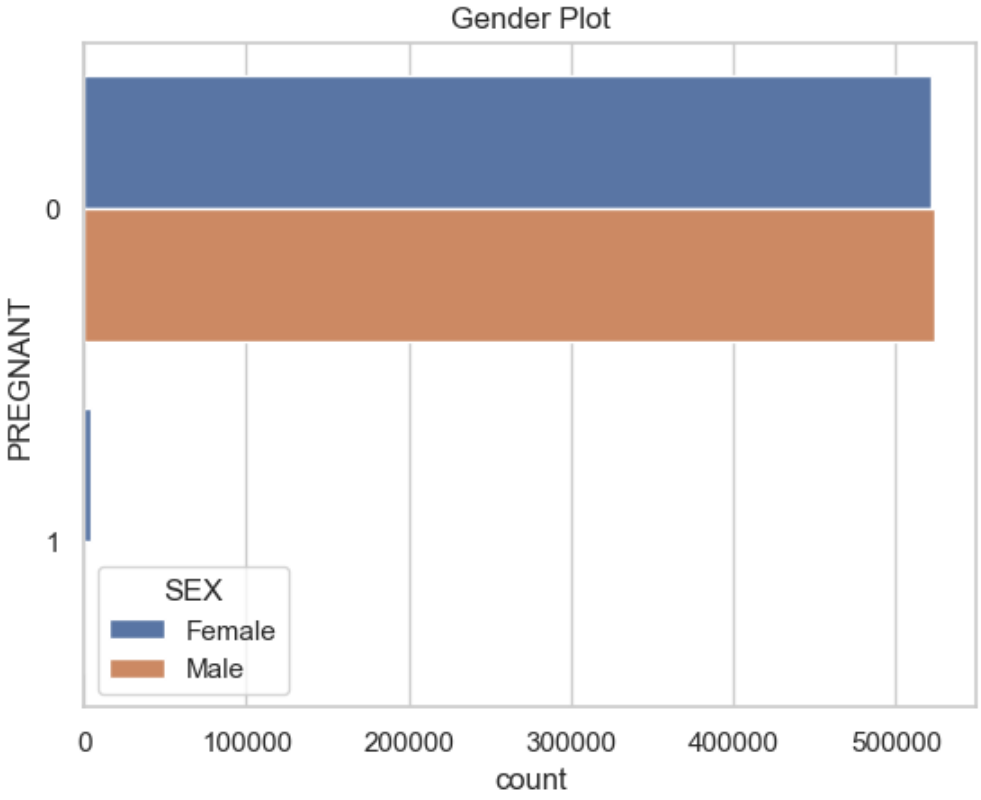
```
repSex = {1: "Female", 2: "Male"}
df.replace({"SEX": repSex},inplace=True)
df['AGE']=change(df['AGE'],[1,11,18,60],["Unknown","Child","Teenager","Adu
repDate={"9999-99-99":0}
df.replace({"DEATH":repDate},inplace=True)
df.loc[df["DEATH"] != 0,"DEATH"]=1
```

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
SEX
Female
Male
INTUBED
0
1
0
100000
200000
300000
400000
count


Gender Plot
PREGNANT
0
1
SEX
Female
Male
0
100000
200000
300000
400000
500000
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

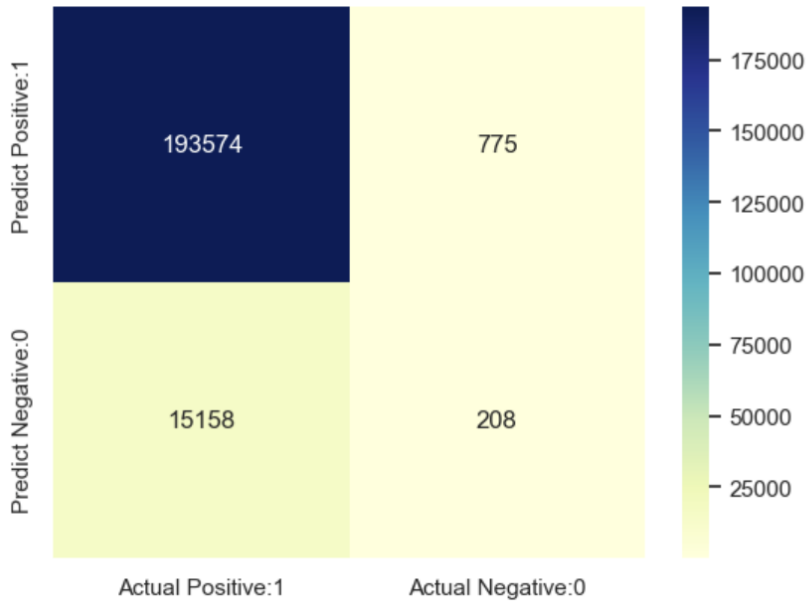
2.2. Kod

```
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["OTHER_DISEASE"].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:  [0.92659562 0.07340438]

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

Score:  0.9240254631285316

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

Training set score: 0.9237
Test set score: 0.9240

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

              precision    recall  f1-score   support

           0       0.93      1.00      0.96    194349
           1       0.21      0.01      0.03     15366

    accuracy                           0.92    209715
   macro avg       0.57      0.50      0.49    209715
weighted avg       0.87      0.92      0.89    209715


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



- Bernoulli

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

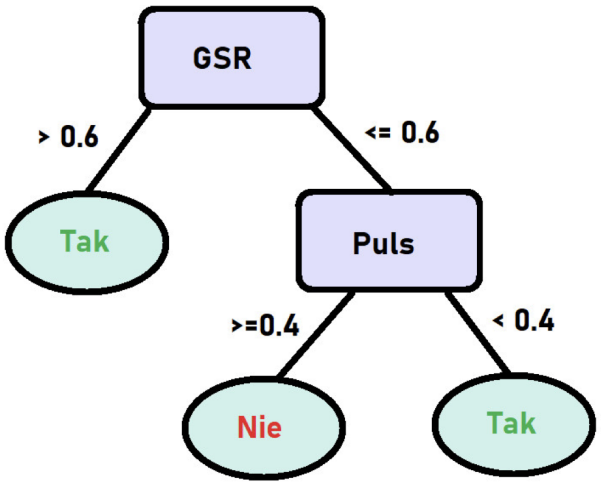
```
Classes:  [0 1]

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

Class Priors:  [-0.07623804 -2.61177164]

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

Score:  0.9267291323939633

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

Training set score: 0.9266
Test set score: 0.9267

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

              precision    recall  f1-score   support

           0       0.93      1.00      0.96    194349
           1       0.00      0.00      0.00     15366

    accuracy                           0.93    209715
   macro avg       0.46      0.50      0.48    209715
weighted avg       0.86      0.93      0.89    209715


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

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

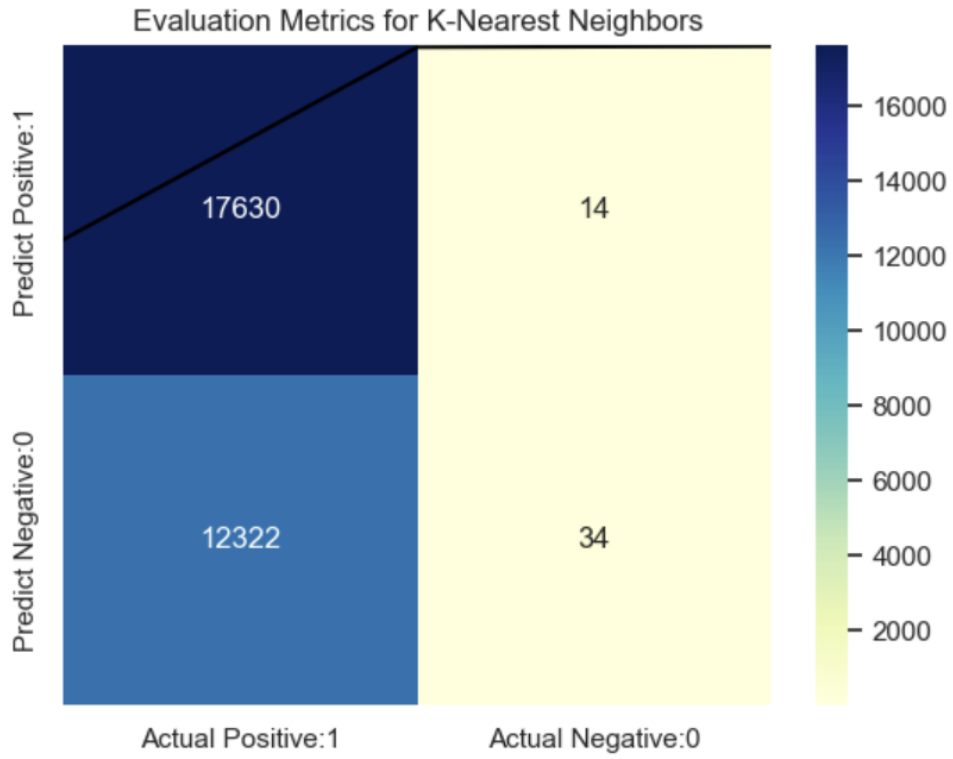
Zbiór testowy

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

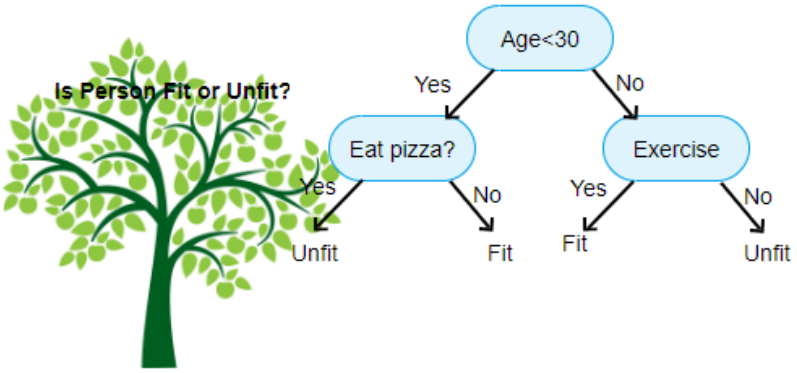
```
              precision    recall  f1-score   support

           0       0.59      1.00      0.74     17644
           1       0.71      0.00      0.01     12356

    accuracy                           0.59     30000
   macro avg       0.65      0.50      0.37     30000
weighted avg       0.64      0.59      0.44     30000

table 1
   0
0  1
1  0
2  1
3  1
4  0
table 2
   0  predictions
0  1            0
1  0            0
2  1            0
3  1            0
4  0            0
0.7083333333333334
0.5888
0.005482102547565302
0.002751699579151829
```



4. Decision-Tree

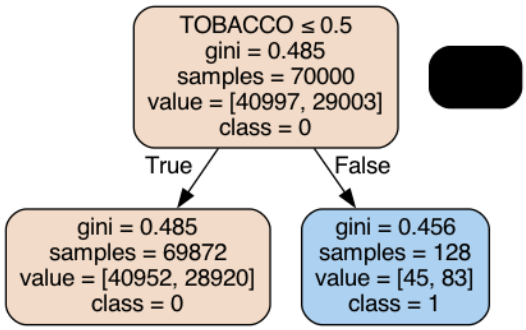
4.1. Definicja

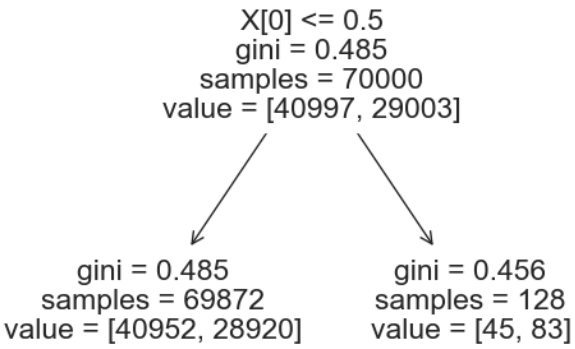
Drzewo decyzyjne jest to jeden ze sposobów klasyfikacji, polegający na podejmowaniu decyzji na podstawie pytań. Przykładem może byc klasyfikacja czy człowiek prowadzi zdrowy tryb życia.



4.2. Kod

```
X=df["OTHER_DISEASE"].iloc[:100000].values.reshape(-1,1)
y = df["DEATH"].iloc[:100000].values.astype("int")
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, r
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())
```

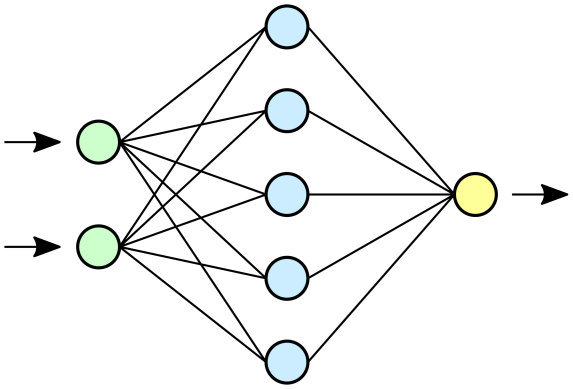




5. Neural-Networks

5.1. Definicja

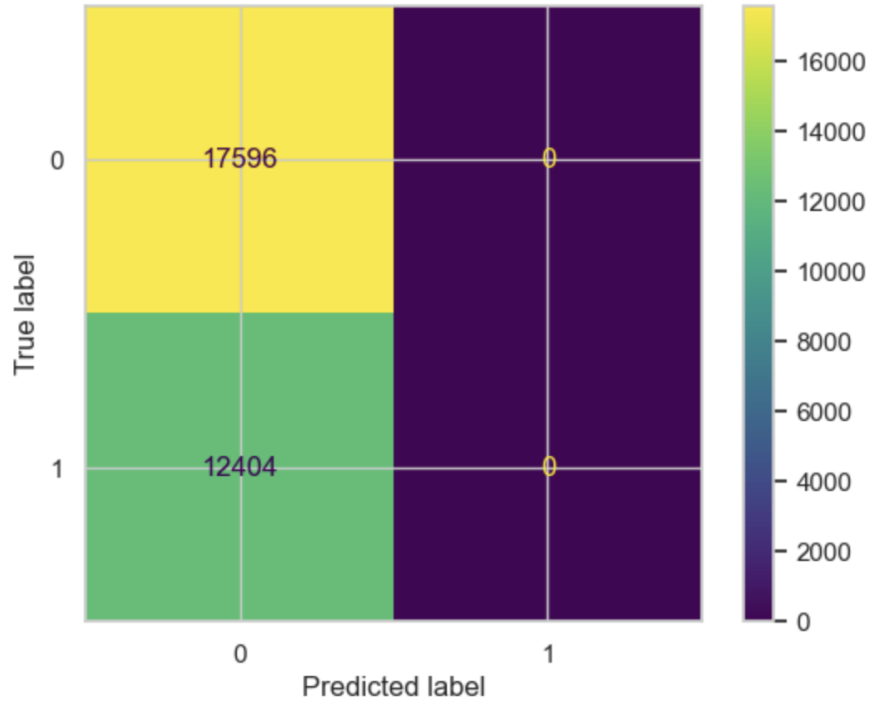
Sieci neuronowe wzorowane są na budowie biologicznego systemu neuronowego w ujęciu matematyczno-informatycznym są grafem skierowanym.



5.2. Kod

```
def neural_network(activation,solver,learning_rate,X,y):
    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=(10, 10), max_iter=1000,
                        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)
    return ["Neural Network", percent, mlp]
r=neural_network('relu','adam','constant',X,y)
plot_confusion_matrix(r[2],X_test,y_test)
```

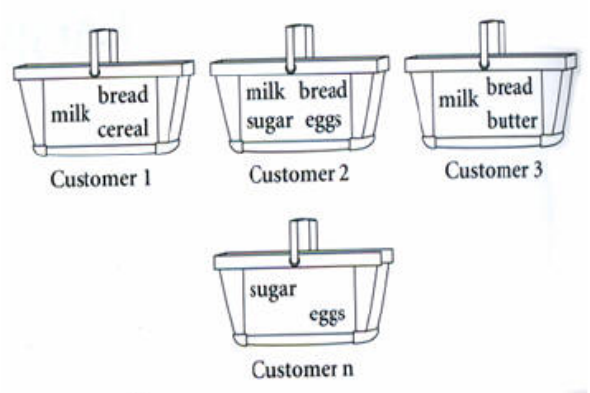
```
[[-0.04280095]
 [-0.04280095]
 [-0.04280095]]
Percent:  0.5872

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x16e194130>
```



6. Apriori

6.1. 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:

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

6.2. Kod

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

```
data = []
df_te=df.iloc[:2000]
df_te['INTUBED']=change(df_te['INTUBED'],[0.5],["NOT INTUBED","INTUBED"])
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('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()
```

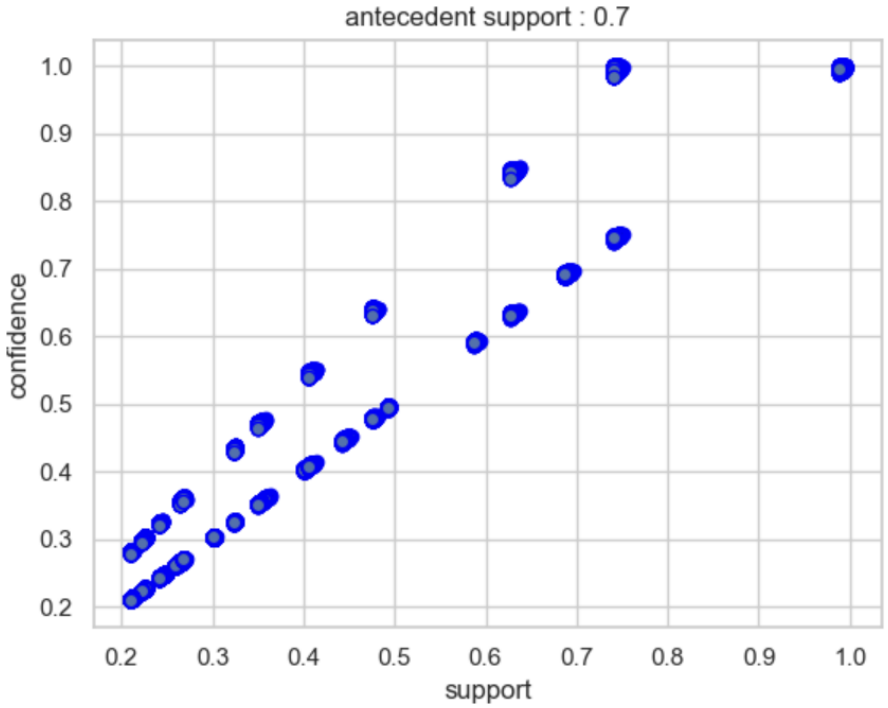
	support	itemsets
0	0.495248	(Adult)
1	0.406203	(Female)
2	0.302151	(INTUBED)
3	0.593797	(Male)
4	0.248624	(NO)

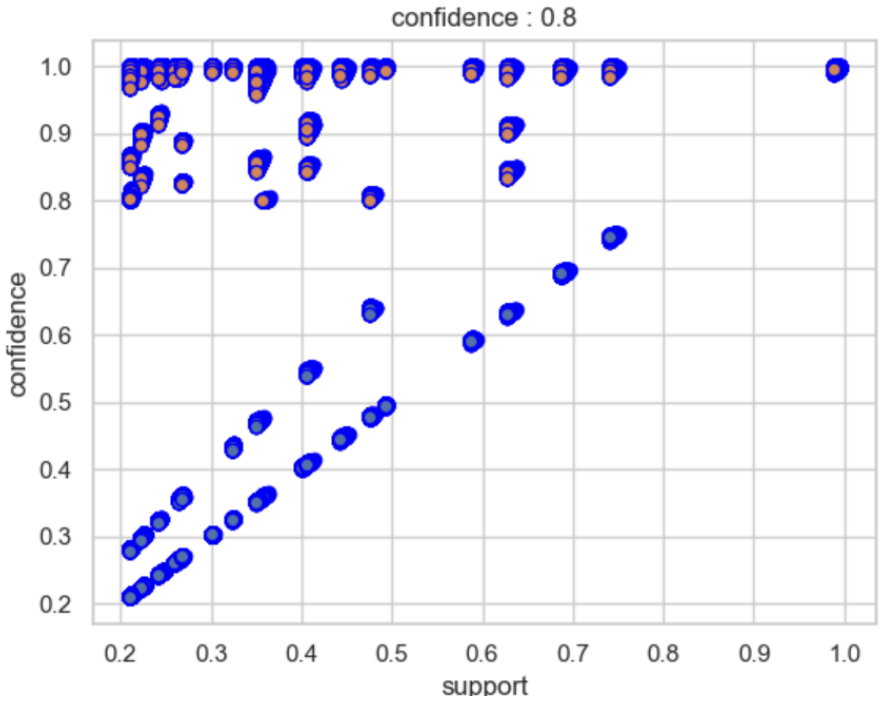
Uruchamianie eksploracji reguł z konfiguracją:

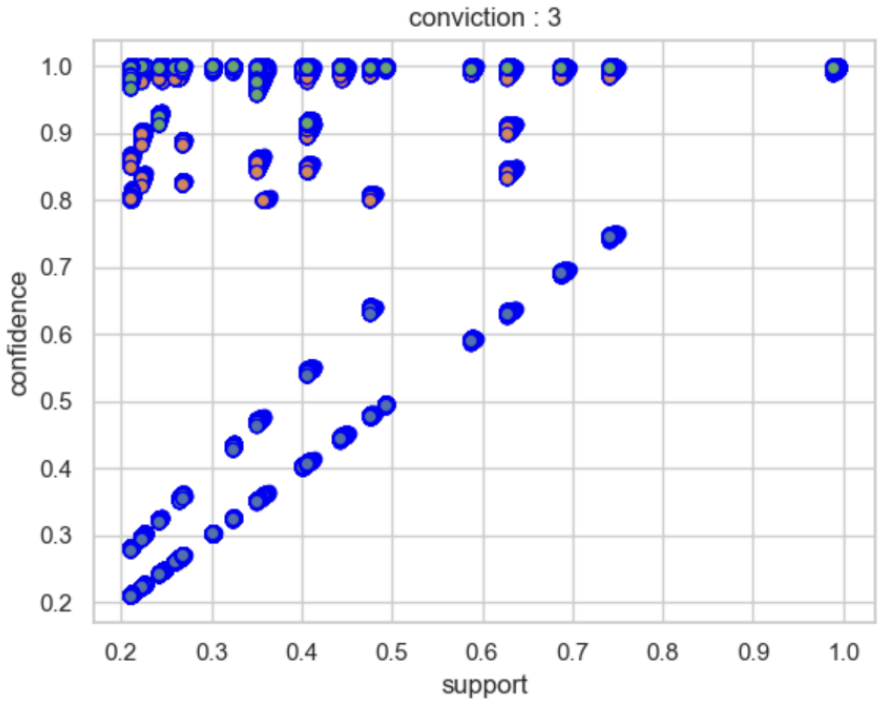
```
config = [ ('antecedent support',0.7),('confidence',0.8),('conviction',3)]
for metric, new_th in config:
    rules = association_rules(apr, metric = metric, min_threshold=new_th)
    if rules.empty:
        print("Dataframe is Empty")
    print(rules.columns.values)
    print("My configuration: ", 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)
...                     ...                                                 ...
28630                (YES)  (NOT OBESITY, NOT PREGNANT, NOT INTUBED, Senio...
28631         (NOT ASTHMA)  (NOT OBESITY, NOT PREGNANT, YES, NOT INTUBED, ...
28632        (NOT DIABETES)  (NOT OBESITY, NOT PREGNANT, YES, NOT INTUBED, ...
28633         (NOT TOBACOO)  (NOT OBESITY, NOT PREGNANT, YES, NOT INTUBED, ...
28634  (NOT OTHER DISEASE)  (NOT OBESITY, NOT PREGNANT, YES, NOT INTUBED, ...

       antecedent support  consequent support   support  confidence      lift  \
0                0.995498            0.495248  0.493747    0.495980  1.001479
1                0.995498            0.495248  0.494247    0.496482  1.002493
2                0.996998            0.495248  0.494247    0.495735  1.000984
3                0.991996            0.495248  0.493747    0.497731  1.005014
4                0.998999            0.495248  0.494747    0.495243  0.999990
...                   ...                 ...       ...         ...       ...
28630            0.751376            0.211606  0.209105    0.278296  1.315161
28631            0.995498            0.209105  0.209105    0.210050  1.004523
28632            0.995498            0.209605  0.209105    0.210050  1.002125
28633            0.993497            0.209605  0.209105    0.210473  1.004144
28634            0.991996            0.210605  0.209105    0.210792  1.000885

       leverage  conviction
0      0.000729    1.001453
1      0.001229    1.002452
2      0.000486    1.000967
3      0.002463    1.004944
4     -0.000005    0.999991
...         ...         ...
28630  0.050109    1.092406
28631  0.000941    1.001197
28632  0.000443    1.000564
28633  0.000863    1.001100
28634  0.000185    1.000236

[28635 rows x 9 columns]
['antecedents' 'consequents' 'antecedent support' 'consequent support'
 'support' 'confidence' 'lift' 'leverage' 'conviction']
My configuration:  confidence  :  0.8
                                        antecedents  \
0                                           (Adult)
1                                           (Adult)
2                                           (Adult)
3                                           (Adult)
4                                           (Adult)
...                                             ...
31761  (Male, Senior, NOT TOBACOO, NOT OTHER DISEASE)
31762                          (Male, Senior, YES)
31763                (NOT INTUBED, Senior, Male)
31764                (Male, Senior, NOT TOBACOO)
31765          (Male, Senior, NOT OTHER DISEASE)
```

```
                                          consequents  antecedent support  \
0                                        (NOT ASTHMA)            0.495248
1                                      (NOT DIABETES)            0.495248
2                                       (NOT OBESITY)            0.495248
3                                  (NOT OTHER DISEASE)            0.495248
4                                      (NOT PREGNANT)            0.495248
...                                               ...                 ...
31761  (NOT OBESITY, NOT PREGNANT, YES, NOT INTUBED, ...            0.259630
31762  (NOT OBESITY, NOT PREGNANT, NOT INTUBED, NOT A...            0.245623
31763  (NOT OBESITY, NOT PREGNANT, YES, NOT ASTHMA, N...            0.216108
31764  (NOT OBESITY, NOT PREGNANT, YES, NOT INTUBED, ...            0.261131
31765  (NOT OBESITY, NOT PREGNANT, YES, NOT INTUBED, ...            0.260630

       consequent support   support  confidence      lift  leverage  \
0                0.995498  0.493747    0.996970  1.001479  0.000729
1                0.995498  0.494247    0.997980  1.002493  0.001229
2                0.996998  0.494247    0.997980  1.000984  0.000486
3                0.991996  0.493747    0.996970  1.005014  0.002463
4                0.998999  0.494747    0.998990  0.999990 -0.000005
...                   ...       ...         ...       ...       ...
31761            0.632816  0.209105    0.805395  1.272715  0.044807
31762            0.686343  0.209105    0.851324  1.240376  0.040523
31763            0.739870  0.209105    0.967593  1.307787  0.049213
31764            0.628814  0.209105    0.800766  1.273454  0.044902
31765            0.631316  0.209105    0.802303  1.270843  0.044565

       conviction
0        1.485743
1        2.228614
2        1.485743
3        2.641321
4        0.990495
...           ...
31761    1.886815
31762    2.109664
31763    8.026871
31764    1.863066
31765    1.864898

[31766 rows x 9 columns]
['antecedents' 'consequents' 'antecedent support' 'consequent support'
 'support' 'confidence' 'lift' 'leverage' 'conviction']
My configuration:  conviction  :  3
                                      antecedents  \
0                                       (INTUBED)
1                                       (INTUBED)
2                                       (INTUBED)
3                                       (INTUBED)
4                                       (INTUBED)
...                                           ...
9965        (NOT INTUBED, Senior, Male, NOT ASTHMA)
9966      (NOT INTUBED, Senior, Male, NOT DIABETES)
9967       (NOT INTUBED, Senior, Male, NOT TOBACOO)
9968  (NOT INTUBED, Senior, Male, NOT OTHER DISEASE)
9969                (NOT INTUBED, Senior, Male)

                                          consequents  antecedent support  \
0                                        (NOT ASTHMA)            0.302151
1                                      (NOT DIABETES)            0.302151
2                                       (NOT OBESITY)            0.302151
3                                  (NOT OTHER DISEASE)            0.302151
4                                       (NOT TOBACOO)            0.302151
```


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
5      (Male)  (NOT PREGNANT)            0.593797            0.998999  0.593797         1.0  1.001002  0.000594         inf

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

     antecedents                                                consequents  antecedent support  consequent support   support  confid
ence      lift  leverage  conviction
30       (Adult)                           (NOT OTHER DISEASE, NOT DIABETES)            0.495248            0.990495  0.493747     0.9
9697  1.006537  0.003206    3.136568
249      (Adult)                (NOT ASTHMA, NOT OTHER DISEASE, NOT DIABETES)            0.495248            0.990495  0.493747     0.9
9697  1.006537  0.003206    3.136568
301      (Adult)                (NOT OBESITY, NOT OTHER DISEASE, NOT DIABETES)            0.495248            0.990495  0.493747     0.9
9697  1.006537  0.003206    3.136568
1157     (Adult)  (NOT OBESITY, NOT OTHER DISEASE, NOT ASTHMA, NOT DIABETES)            0.495248            0.990495  0.493747     0.9
9697  1.006537  0.003206    3.136568

------------------------------------------
```

7. Link do githuba

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

8. 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