Uniwersytet Gdański
Wydział Matematyki, Fizyki i Informatyki

Informatyka Ogólnoakademicka

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/einsteindata4u/covid19`

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
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. 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
```

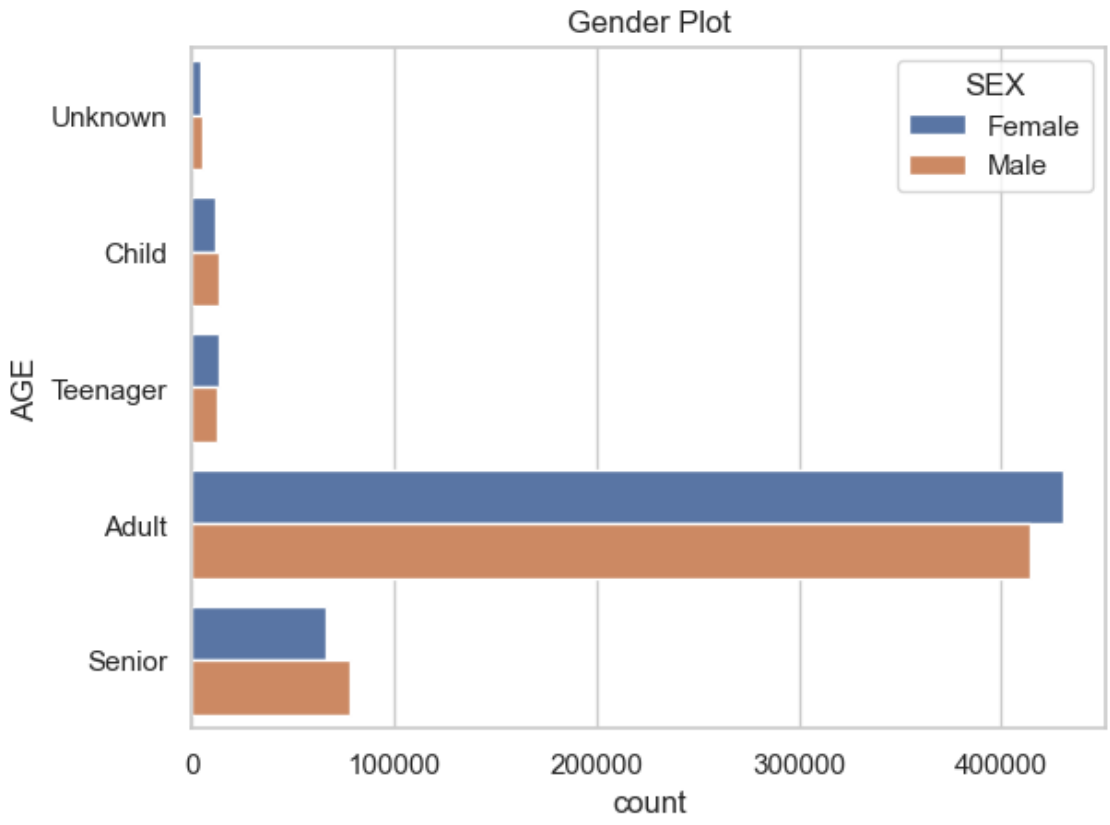
1.4. Dane na wykresach

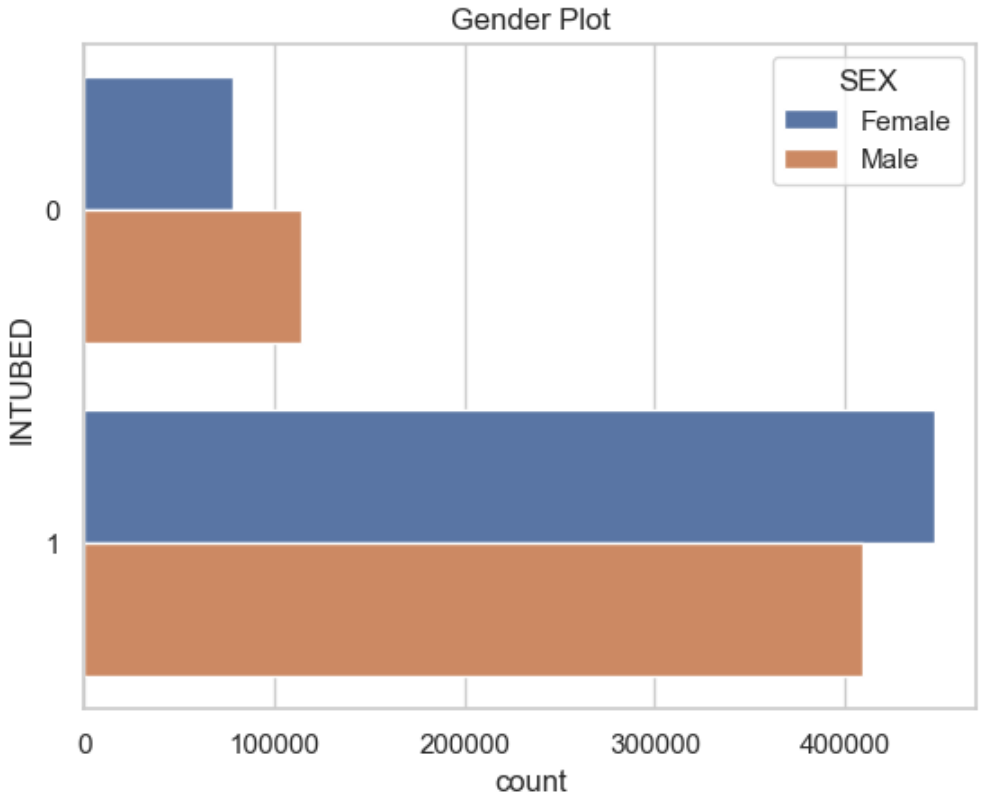
Aby baza danych była bardziej czytelna dla użytkownika została lekko zmodyfikowana.

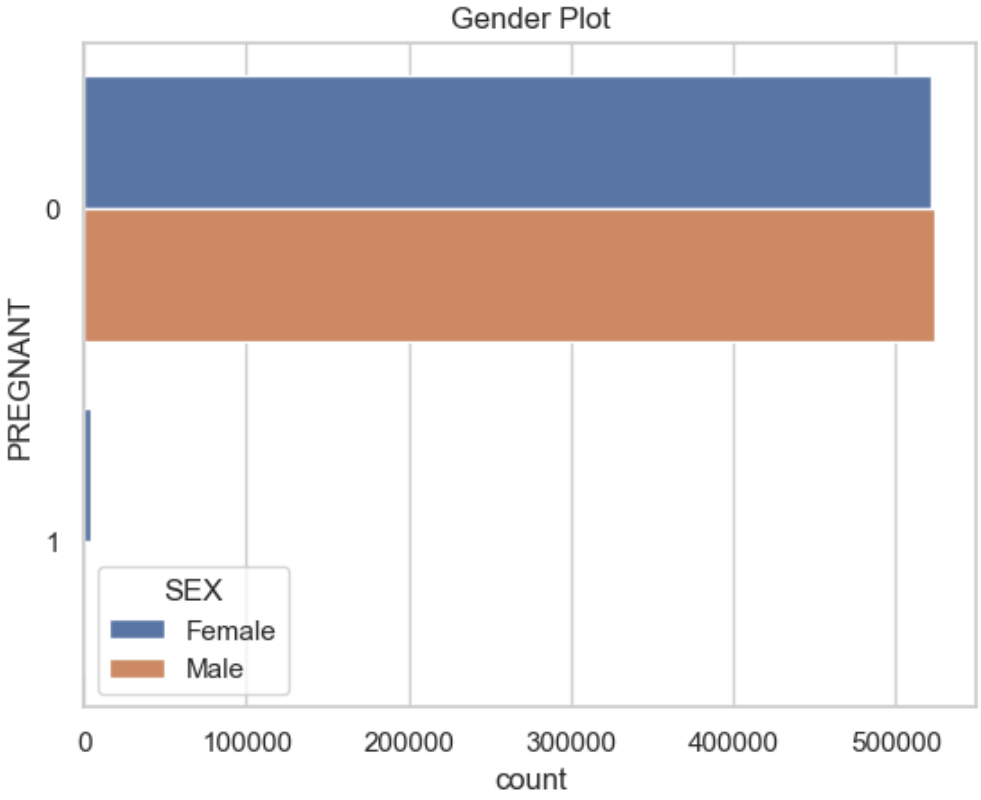
```
repSex = {1: "Female", 2: "Male"}
df.replace({"SEX": repSex},inplace=True)
df['AGE']=change(df['AGE'],[1,11,18,60],["Unknown","Child","Teenager","Adu
repDate={"9999-99-99":0}
df.replace({"DEATH":repDate},inplace=True)
df.loc[df["DEATH"] != 0,"DEATH"]=1
```

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 B prawdziwe

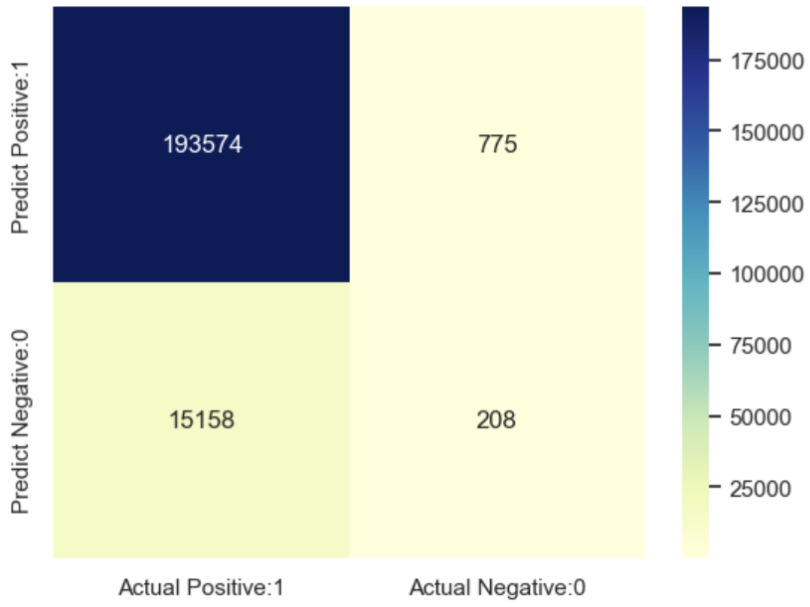
2.2. Kod

```
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
```

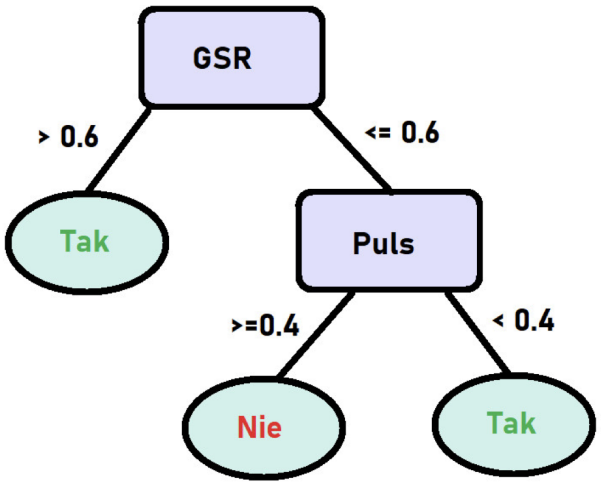
```
Classes:  [0 1]

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

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

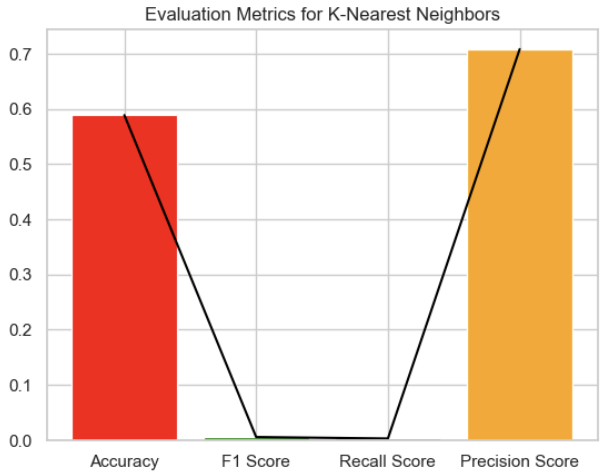
Zbiór testowy

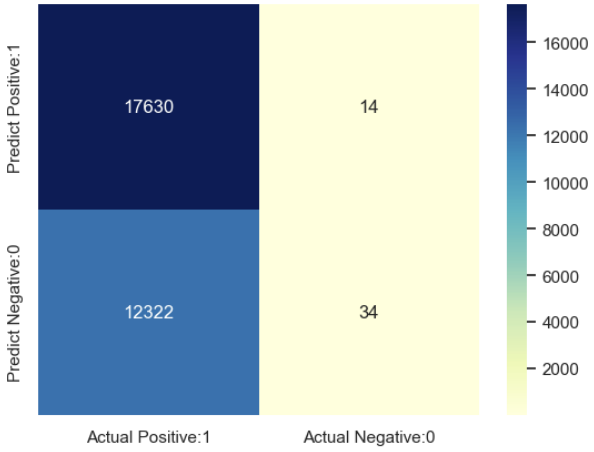
Puls	GSR	Winny
0,4	0,6	Nie
0,6	0,6	Tak
0,4	0,9	Tak
0,5	0,2	Nie
0,5	0,6	Tak

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()
    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
```

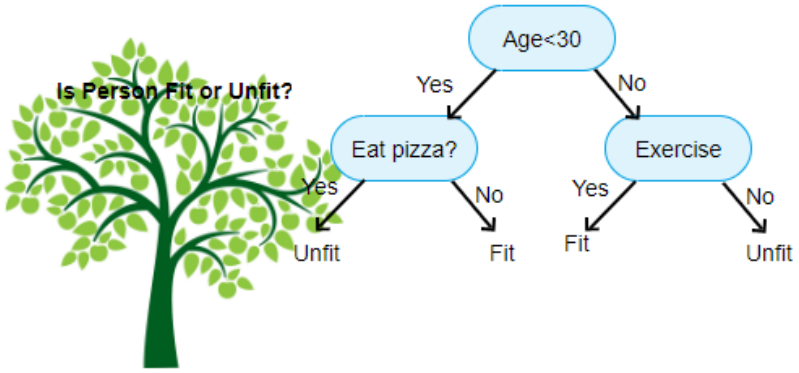




4. Decision-Tree

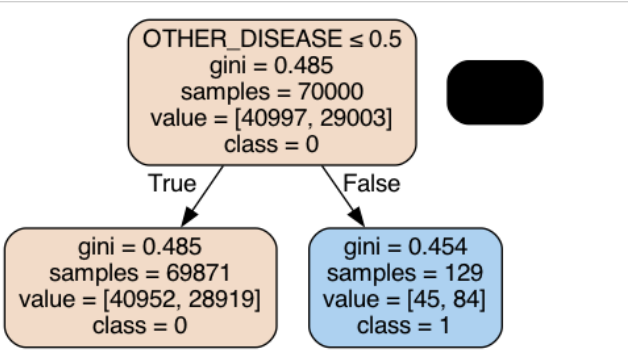
4.1. Definicja

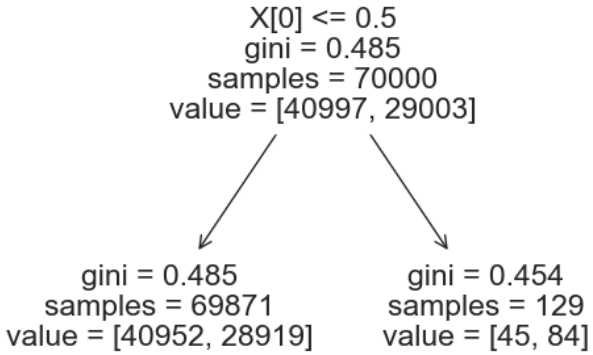
Drzewo decyzyjne jest to jeden ze sposobów klasyfikacji, polegający na podejmowaniu decyzji na podstawie pytań. Przykładem może byc klasyfikacja czy człowiek zdrowo się odżywia.



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,
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())
```

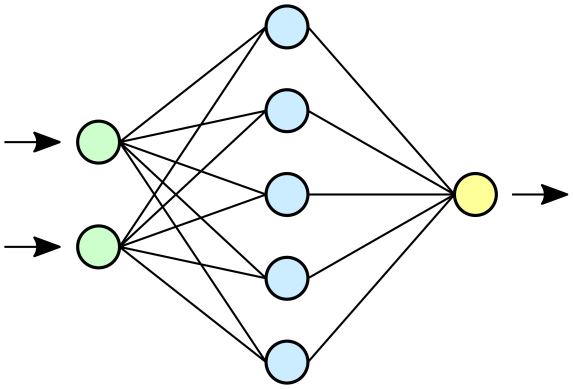




5. Neural-Networks

5.1. Definicja

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

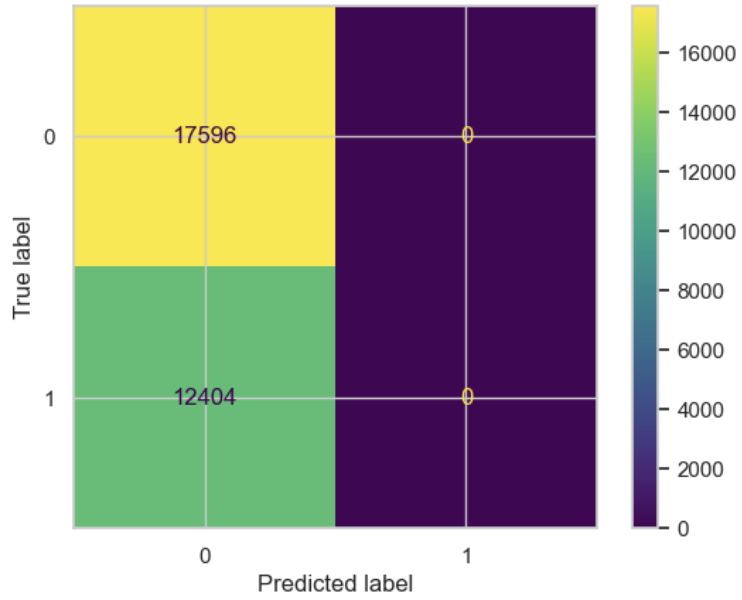


5.2. Kod

```
def neural_network():
    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)

    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))

    return ["Neural Network", percent, mlp]

r=neural_network()
plot_confusion_matrix(r[2],X_test,y_test)
```

```
[[-0.04280095]
 [-0.04280095]
 [-0.04280095]]
```

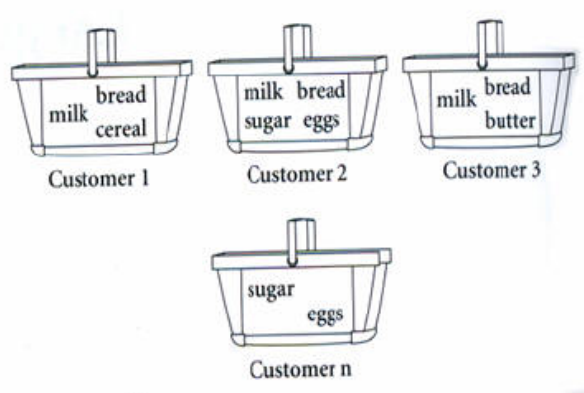
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x177f7f8b0>
```



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[:1000]
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()
```

	0	1	2	3	Adult	Child	False	Female	Male	Senior	Teenager	True	Unknown
0	False	True	True	False	False	False	True	True	False	True	False	True	False
1	False	True	True	False	False	False	True	False	True	True	False	True	False
2	False	True	True	False	True	False	True	False	True	False	False	False	False
3	False	True	True	False	True	False	True	True	False	False	False	True	False
4	False	True	True	False	False	False	True	False	True	True	False	True	False

Wyniki aprori:

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

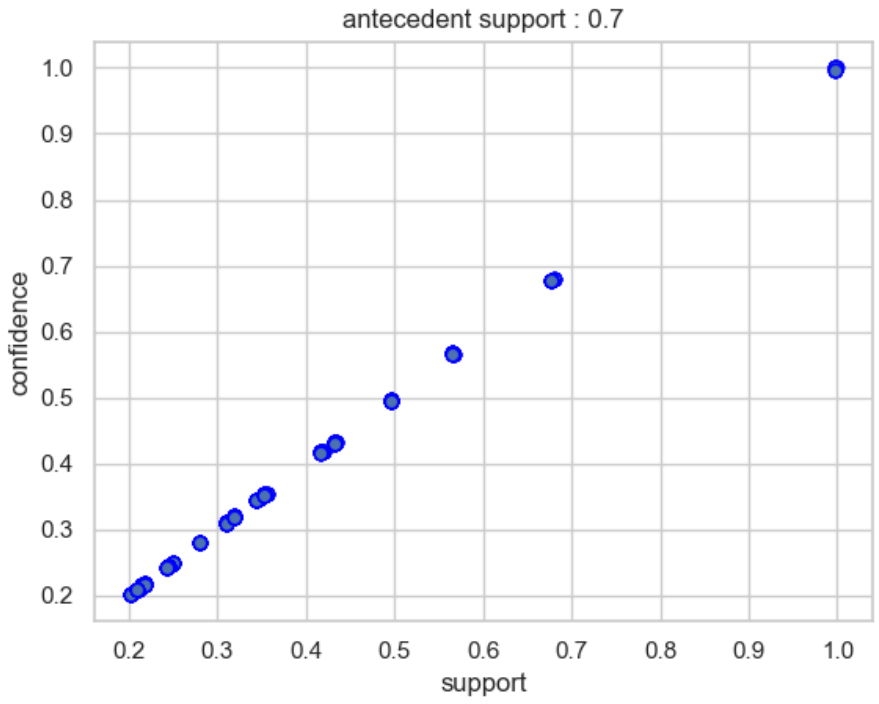
	support	itemsets
0	0.310310	(0)
1	1.000000	(1)
2	0.996997	(2)
3	0.679680	(3)
4	0.495495	(Adult)

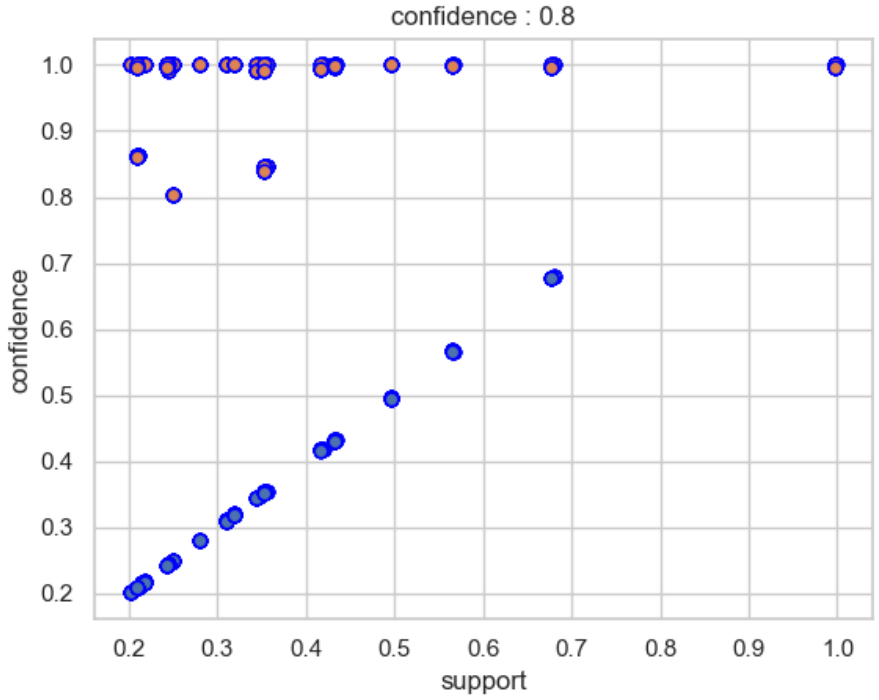
Uruchamianie eksploracji reguł z konfiguracją:

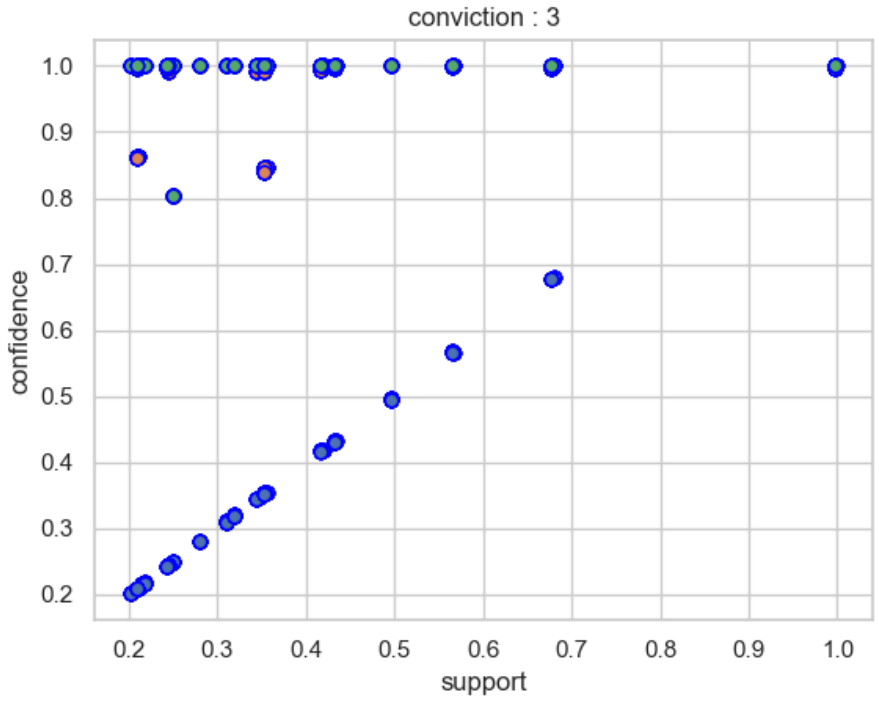
```
config = [ ('antecedent support',0.7),('confidence',0.8),('conviction',3)]
for metric, new_th in config:
    rules = association_rules(apr, metric = metric, min_threshold=new_th)
    if rules.empty:
        print("Dataframe is Empty")
    print(rules.columns.values)
    print("My configuration: ", 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  antecedent support  \
0           (1)                         (0)            1.000000
1           (2)                         (0)            0.996997
2       (False)                         (0)            1.000000
3           (2)                         (1)            0.996997
4           (1)                         (2)            1.000000
..          ...                         ...                 ...
349  (1, False)         (2, Senior, Male, 3)            1.000000
350  (2, False)         (Senior, 1, Male, 3)            0.996997
351         (1)  (2, Male, Senior, False, 3)            1.000000
352         (2)  (1, Male, Senior, False, 3)            0.996997
353     (False)      (1, 2, Male, Senior, 3)            1.000000

     consequent support   support  confidence      lift  leverage  conviction
0              0.310310  0.310310    0.310310  1.000000  0.000000    1.000000
1              0.310310  0.310310    0.311245  1.003012  0.000932    1.001357
2              0.310310  0.310310    0.310310  1.000000  0.000000    1.000000
3              1.000000  0.996997    1.000000  1.000000  0.000000         inf
4              0.996997  0.996997    0.996997  1.000000  0.000000    1.000000
..                  ...       ...         ...       ...       ...         ...
349            0.209209  0.209209    0.209209  1.000000  0.000000    1.000000
350            0.210210  0.209209    0.209839  0.998236 -0.000370    0.999531
351            0.209209  0.209209    0.209209  1.000000  0.000000    1.000000
352            0.210210  0.209209    0.209839  0.998236 -0.000370    0.999531
353            0.209209  0.209209    0.209209  1.000000  0.000000    1.000000

[354 rows x 9 columns]
['antecedents' 'consequents' 'antecedent support' 'consequent support'
 'support' 'confidence' 'lift' 'leverage' 'conviction']
My configuration:  confidence  :  0.8
              antecedents       consequents  antecedent support  \
0                     (0)               (1)            0.310310
1                     (0)               (2)            0.310310
2                     (0)           (False)            0.310310
3                     (0)            (True)            0.310310
4                     (2)               (1)            0.996997
..                    ...               ...                 ...
430     (Senior, 1, Male)     (2, False, 3)            0.243243
431     (2, Senior, Male)     (1, False, 3)            0.242242
432  (Senior, False, Male)         (2, 1, 3)            0.243243
433     (Senior, Male, 3)     (2, 1, False)            0.210210
434        (Senior, Male)  (2, 1, False, 3)            0.243243

     consequent support   support  confidence      lift  leverage  conviction
0              1.000000  0.310310    1.000000  1.000000  0.000000         inf
1              0.996997  0.310310    1.000000  1.003012  0.000932         inf
2              1.000000  0.310310    1.000000  1.000000  0.000000         inf
3              0.347347  0.249249    0.803226  2.312457  0.141464    3.316759
4              1.000000  0.996997    1.000000  1.000000  0.000000         inf
..                  ...       ...         ...       ...       ...         ...
430            0.676677  0.209209    0.860082  1.271039  0.044612    2.310811
431            0.679680  0.209209    0.863636  1.270652  0.044562    2.349016
432            0.676677  0.209209    0.860082  1.271039  0.044612    2.310811
433            0.996997  0.209209    0.995238  0.998236 -0.000370    0.630631
434            0.676677  0.209209    0.860082  1.271039  0.044612    2.310811
```

```
[435 rows x 9 columns]
['antecedents' 'consequents' 'antecedent support' 'consequent support'
 'support' 'confidence' 'lift' 'leverage' 'conviction']
My configuration:  conviction  :  3
                    antecedents    consequents  antecedent support  \
0                           (0)            (1)            0.310310
1                           (0)            (2)            0.310310
2                           (0)        (False)            0.310310
3                           (0)         (True)            0.310310
4                           (2)            (1)            0.996997
..                          ...            ...                 ...
281      (False, Adult, Male, 3)         (2, 1)            0.217217
282            (Adult, Male, 3)  (2, 1, False)            0.217217
283      (1, 2, Male, Senior, 3)        (False)            0.209209
284  (2, Male, Senior, False, 3)            (1)            0.209209
285         (2, Senior, Male, 3)     (1, False)            0.209209

     consequent support   support  confidence      lift  leverage  conviction
0              1.000000  0.310310    1.000000  1.000000  0.000000         inf
1              0.996997  0.310310    1.000000  1.003012  0.000932         inf
2              1.000000  0.310310    1.000000  1.000000  0.000000         inf
3              0.347347  0.249249    0.803226  2.312457  0.141464    3.316759
4              1.000000  0.996997    1.000000  1.000000  0.000000         inf
..                  ...       ...         ...       ...       ...         ...
281            0.996997  0.217217    1.000000  1.003012  0.000652         inf
282            0.996997  0.217217    1.000000  1.003012  0.000652         inf
283            1.000000  0.209209    1.000000  1.000000  0.000000         inf
284            1.000000  0.209209    1.000000  1.000000  0.000000         inf
285            1.000000  0.209209    1.000000  1.000000  0.000000         inf
```




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


conviction : 3
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({'Female'})].to_string())
print("\n------------------------------------------\n")
print(rules[rules['antecedents']==frozenset({'Male'})].to_string())
print("\n------------------------------------------\n")
print(rules[rules['antecedents']==frozenset({'Senior'})].to_string())
print("\n------------------------------------------\n")
```

```
   antecedents consequents  antecedent support  consequent support   support  confidence  lift  lev
erage  conviction
5     (Female)         (0)            0.433433                 1.0  0.433433         1.0   1.0
0.0         inf
11    (Female)         (1)            0.433433                 1.0  0.433433         1.0   1.0
0.0         inf
26    (Female)      (0, 1)            0.433433                 1.0  0.433433         1.0   1.0
0.0         inf

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

   antecedents consequents  antecedent support  consequent support   support  confidence  lift  lev
erage  conviction
6       (Male)         (0)            0.566567                 1.0  0.566567         1.0   1.0
0.0         inf
12      (Male)         (1)            0.566567                 1.0  0.566567         1.0   1.0
0.0         inf
29      (Male)      (0, 1)            0.566567                 1.0  0.566567         1.0   1.0
0.0         inf

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

   antecedents consequents  antecedent support  consequent support   support  confidence  lift  lev
erage  conviction
7     (Senior)         (0)            0.419419                 1.0  0.419419         1.0   1.0
0.0         inf
13    (Senior)         (1)            0.419419                 1.0  0.419419         1.0   1.0
0.0         inf
32    (Senior)      (0, 1)            0.419419                 1.0  0.419419         1.0   1.0
0.0         inf

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

7. Link do githuba

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

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