Neuronowy model klienta bankowego (Bank Marketing Data)

KWD

Katarzyna Szpila

06.02.2020

Streszczenie

Spis treści

Rozdział 1

Wprowadzenie

Dane wykorzystane w projekcie pochodzą z kampanii marketingowej depozytów portugalskiej instytucji bankowej. Problemem biznesowym jest problem klasyfikacji binarnej. Celem klasyfikacji jest przewidzenie, czy klient skontaktowany za pośrednictwem kampanii marketingowej zasubskrybuje lokatę terminową. Będę próbowała ten probelm rozwiązać tworząc odpowiedni model sieci neuronowej.

Rozdział 2

Opis danych wejściowych

Dane pochodzą ze strony: *https://www.kaggle.com/janiobachmann/bank-marketing-dataset.*
Są one powiązane z kampaniami marketingu portugalskiej instytucji bankowej. Kampanie marketingowe opierały się na rozmowach telefonicznych. Często wymagany był więcej niż jeden kontakt z tym samym klientem, aby uzyskać dostęp, jeśli produkt (lokata bankowa) będzie (lub nie) subskrybowany.
Zestaw danych ma 17 atrybutów, w tym jeden atrybut zależny, i istnieje 45211 instancji danych. Mamy więc 16 atrybutów niezależnych i 1 atrybut zależny.
Zawierają następujące kolumny:

Atrybuty klienta bankowego:
wiek: wiek klienta (numerycznie)
- praca: rodzaj pracy (kategorie: „admin.", „nieznany", „bezrobotny", „zarządzanie", „pokojówka", „przedsiębiorca", „student", „pracownik fizyczny", „samozatrudniony", „ na emeryturze ",„ technik ",„ usługi ")
- stan cywilny: stan cywilny (kategorie: „żonaty", „rozwiedziony", „samotny")
-edukacja: najwyższe wykształcenie klienta (kategorie: „nieznane", „średnie", „podstawowe", „wyższe")
- default: czy kredyt jest domyślnie? (binarne kategorie: „tak", „nie")
- saldo: średni roczny bilans, w euro (numeryczny)
- mieszkanie: czy ma kredyt mieszkaniowy? (binarne kategorie: „tak", „nie")
-pożyczka: czy pożyczka osobista? (binarne kategorie: „tak", „nie")

Związane z ostatnim kontaktem bieżącej kampanii:
-kontakt: typ komunikacji kontaktowej (kategorie: „nieznany", „telefon", „komórkowy")

-dzień: dzień ostatniego kontaktu miesiąca (numerycznie)
- miesiąc: ostatni kontakt miesiąc roku (kategorie: „sty", „lut", „mar", ..., „lis", „gru")
- czas trwania: czas trwania ostatniego kontaktu, w sekundach (numerycznie)

Inne atrybuty:
- kampania: liczba kontaktów wykonanych podczas tej kampanii i dla tego klienta (numerycznie, obejmuje ostatni kontakt)
-pdays: liczba dni, które upłynęły od ostatniego kontaktu klienta z poprzedniej kampanii (wartość liczbowa, -1 oznacza, że
klient -wcześniej się nie skontaktował)
-poprzedni: liczba kontaktów wykonanych przed tą kampanią i dla tego klienta (numerycznie)
-poutcome: wynik poprzedniej kampanii marketingowej (kategorie: „nieznany", „inny", „porażka", „sukces")

Zmienna wyjściowa (pożądany cel):
-deposit: czy klient subskrybował lokatę terminową? (binarnie: „tak", „nie")

opis danych wejściowych i ich analiza eksploracyjna/wizualizacja - W razie konieczności normalizacja lub standaryzacja danych (powinno to zostać uzasadnione)
Poniższa tabela przedstawia kolumny i przykładowe dane w wierszach:

	0	1	2	3	4
age	59	56	41	55	54
job	admin.	admin.	technician	services	admin.
marital	married	married	married	married	married
education	secondary	secondary	secondary	secondary	tertiary
default	no	no	no	no	no
balance	2343	45	1270	2476	184
housing	yes	no	yes	yes	no
loan	no	no	no	no	no
contact	unknown	unknown	unknown	unknown	unknown
day	5	5	5	5	5
month	may	may	may	may	may
duration	1042	1467	1389	579	673
campaign	1	1	1	1	2
pdays	-1	-1	-1	-1	-1
previous	0	0	0	0	0
poutcome	unknown	unknown	unknown	unknown	unknown
deposit	yes	yes	yes	yes	yes

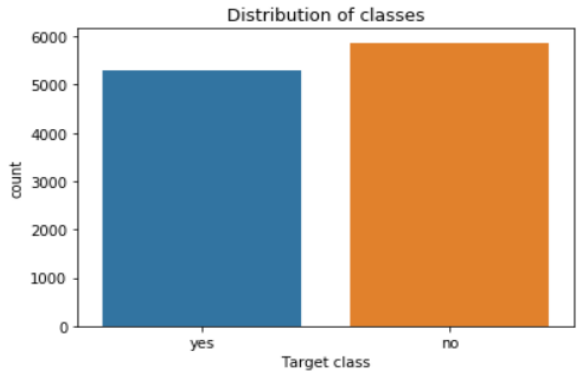
Ddatkowe informacje o naszych danych:

	age	balance	day	duration	campaign	pdays	previous
count	11162.000000	11162.000000	11162.000000	11162.000000	11162.000000	11162.000000	11162.000000
mean	41.231948	1528.538524	15.658036	371.993818	2.508421	51.330407	0.832557
std	11.913369	3225.413326	8.420740	347.128386	2.722077	108.758282	2.292007
min	18.000000	-6847.000000	1.000000	2.000000	1.000000	-1.000000	0.000000
25%	32.000000	122.000000	8.000000	138.000000	1.000000	-1.000000	0.000000
50%	39.000000	550.000000	15.000000	255.000000	2.000000	-1.000000	0.000000
75%	49.000000	1708.000000	22.000000	496.000000	3.000000	20.750000	1.000000
max	95.000000	81204.000000	31.000000	3881.000000	63.000000	854.000000	58.000000

Możemy z nich odczytać wartości minimalne, maksymalne, średnie dla danych kolumn oraz podliczone wszystkie wartości. Typy danych w kolumnach wyglądają następująco:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11162 entries, 0 to 11161
Data columns (total 17 columns):
age          11162 non-null int64
job          11162 non-null object
marital      11162 non-null object
education    11162 non-null object
default      11162 non-null object
balance      11162 non-null int64
housing      11162 non-null object
loan         11162 non-null object
contact      11162 non-null object
day          11162 non-null int64
month        11162 non-null object
duration     11162 non-null int64
campaign     11162 non-null int64
pdays        11162 non-null int64
previous     11162 non-null int64
poutcome     11162 non-null object
deposit      11162 non-null object
dtypes: int64(7), object(10)
memory usage: 1.4+ MB
```

Analizując rozmieszczenie danych w zmiennej deposit, którą badamy możemy wywnioskować że dane są zbilansowane.



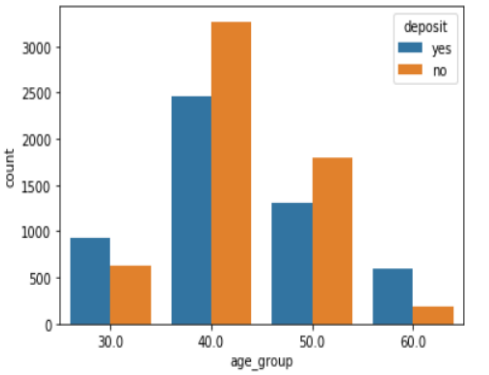
Liczbowo wygląda to następująco: no 5873 yes 5289

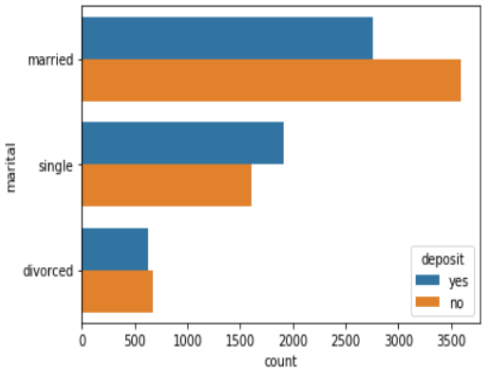
Redukcja wymiarowości:
Analiza danych wykazuje, że zmiennna pdays, oznaczająca liczba dni, które upłynęły od ostatniego kontaktu klienta z poprzed-niej kampanii (wartość liczbowa, -1 oznacza, że klient -wcześniej się nie skontaktował), oraz poutcome oznaczająca wynik poprzedniej kampanii marketingowej (kategorie: „nieznany", „inny", „porażka", „sukces"), nie wnoszą dużo do naszego modelu. Zatem z nich zrezygnujemy. Poniższe statystyki pokazują rozmieszczenie danych w tych kolumnach:

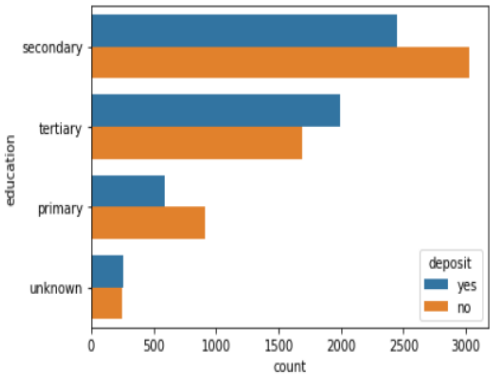
```
In [6]: df.pdays.value_counts()

Out[6]: -1      8324
         92      106
         182      89
         91       84
         181      81
                 ...
         587       1
         579       1
         515       1
         491       1
         683       1
        Name: pdays, Length: 472, dtype: int64
```

```
In [8]: df.poutcome.value_counts()

Out[8]: unknown    8326
        failure    1228
        success    1071
        other       537
        Name: poutcome, dtype: int64
```

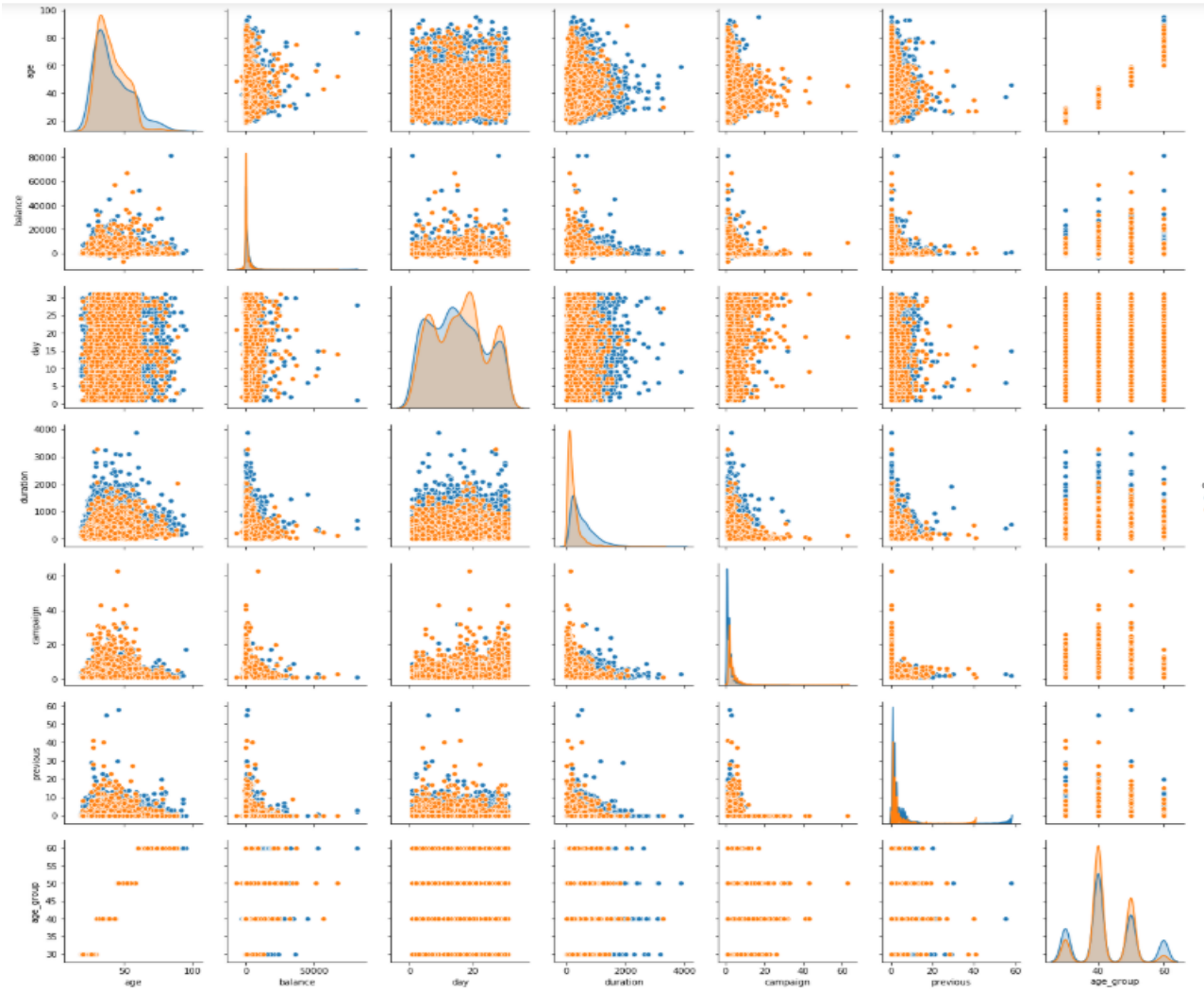
Dodatkowe statystyki z analizy danych w celu badania korelacji:







Macierz korelacji:



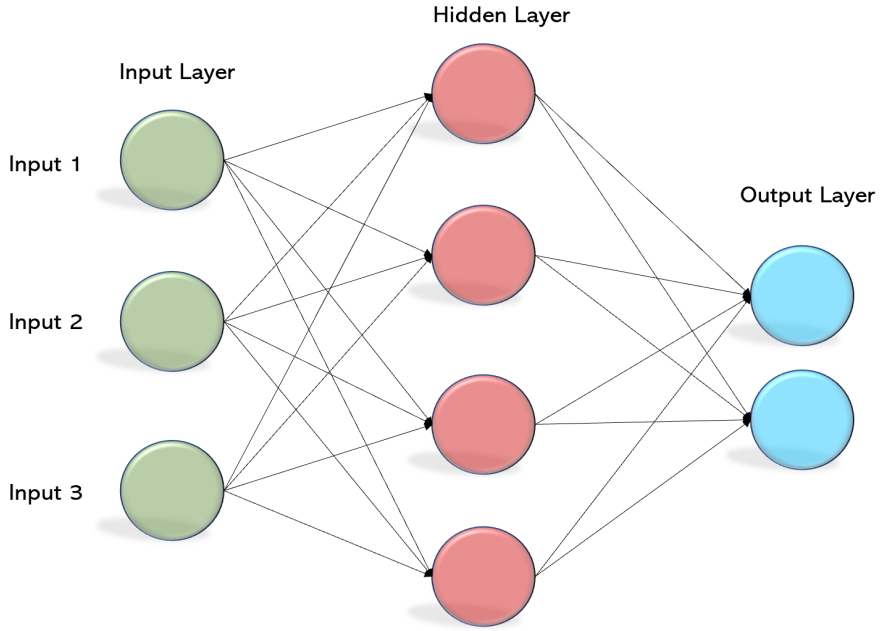
Analizując powyższe wyniki, doszłam do wnioski że w większości dane są skorelowane, zatem nie ma już potrzeby redukcji wymiarów.

Nasze dane są w postaci liczb i opisów zatem ustandaryzujemy je używając preprocessing.LabelEncoder() z biblioteki sklearn. W wyniku tej operacji wszystkie nasze dane będą liczbowe.

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	deposit
0	59	0	1	1	0	2343	1	0	2	5	8	1042	1	-1	0	3	1
1	56	0	1	1	0	45	0	0	2	5	8	1467	1	-1	0	3	1
2	41	9	1	1	0	1270	1	0	2	5	8	1389	1	-1	0	3	1
3	55	7	1	1	0	2476	1	0	2	5	8	579	1	-1	0	3	1
4	54	0	1	2	0	184	0	0	2	5	8	673	2	-1	0	3	1

2.1 Wprowadzenie teoretyczne

Uczenie sieci MLP to optymalizacja wartościwag w celu minimalizacji błędu popełnianego przez sieć.
Multi-layer Perceptron (MLP) to nadzorowany algorytm uczenia, który uczy się funkcji poprzez szkolenie na zbiorze danych. Może nauczyć się aproksymować funkcje nieliniowe do klasyfikacji lub regresji. Różni się od regresji logistycznej tym, że między warstwą wejściową i wyjściową może znajdować się jedna lub więcej warstw nieliniowych, zwanych warstwami ukrytymi.
Zalety MLP:
-jest w stanie przybliżyć dowolnie złożoone i skomplikowane odwzorowanie
-użytkownik nie musi znać lub zakłada z góry żadnej formy występujących w poszukiwanym modelu zależności
-nie musi nawet zadawa sobie pytania, czy jakiekolwiek możliwe do matematycznego modelowania zależności w ogóle występują
-wygodne narzędzie do wszelkiego rodzaju zastosowań związanych z prognozowaniem, klasyfkacj¡ lub automatycznym sterowaniem



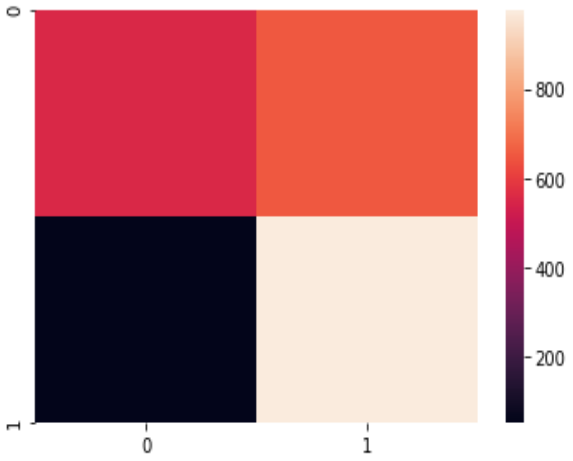
Rysunek 1 pokazuje jedną ukrytą warstwę MLP z wyjściem skalarnym.

2.2 Badania symulacyjne

Analiza jakości uzyskanego wyniku:
W pierwszym modelu poprawność symulacji na zbiorze testowym kształtuje sie na poziomie 68%.

```
neural_network = MLPClassifier(hidden_layer_sizes=(200,100,50), activation = 'relu',\
                               solver = 'adam', random_state=1, verbose = False, learning_rate='adaptive')
neural_network.fit(x_train, y_train.ravel())

pr = neural_network.predict(x_test)
from sklearn.metrics import confusion_matrix
conf_matrix_neural_network = confusion_matrix(y_test.ravel(),pr)
print("Confusion_matrix:")
print(conf_matrix_neural_network)
import seaborn as sns;
sns.heatmap(conf_matrix_neural_network)
acc = accuracy_score(y_test.ravel(),pr)

print("Neural network model accuracy is {0:0.2f}".format(acc))
```

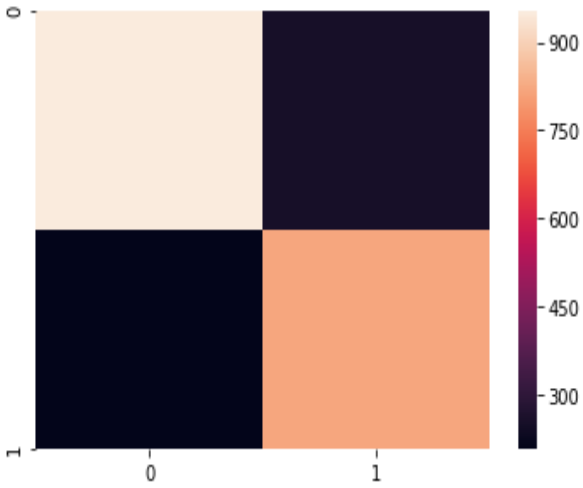
```
Confusion_matrix:
[[554 651]
 [ 53 975]]
Neural network model accuracy is 0.68
```



W celu polepszenia rozwiązania eksperymentalnie zmieniałam ilośc powłok, a także parametry "solver" oraz "activation" ostatecznie otrzymując 79% poprawności zakwalifikowania na zbiorze testowym:

```
neural_network = MLPClassifier(hidden_layer_sizes=(40,40,40), activation = 'logistic',\
                               solver = 'adam',  random_state=1, verbose = False)
neural_network.fit(x_train, y_train.ravel())

pr = neural_network.predict(x_test)
from sklearn.metrics import confusion_matrix
conf_matrix_neural_network = confusion_matrix(y_test.ravel(),pr)
print("Confusion_matrix:")
print(conf_matrix_neural_network)
import seaborn as sns;
sns.heatmap(conf_matrix_neural_network)
acc = accuracy_score(y_test.ravel(),pr)

print("Neural network model accuracy is {0:0.2f}".format(acc))
```

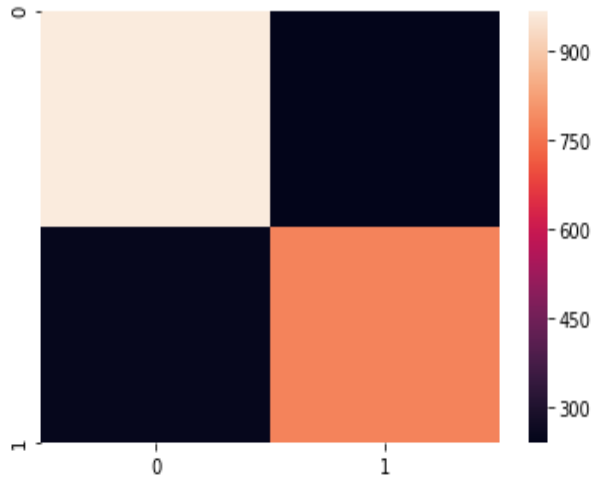
```
Confusion_matrix:
[[952 253]
 [208 820]]
Neural network model accuracy is 0.79
```



2.3 Porównanie z innymi metodami

Porównanie wyników z symulacjami na innych modelach:
Wynik macierzy konfuzji w modelu regresji liniowej, gdzie poprawnośc kształtuje na poziomie 78%:

```
[[965 240]
 [250 778]]

<matplotlib.axes._subplots.AxesSubplot at 0x27ea4de4ec8>
```

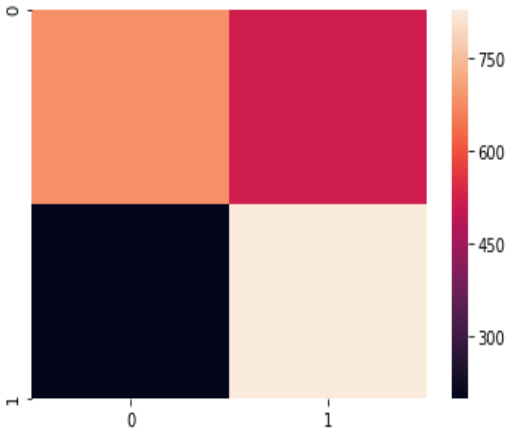


Wynik modelu pojedynczego neuronu:

```
ppn = Perceptron(max_iter = 77, verbose=0, penalty="elasticnet", random_state = 1, tol=None)
ppn.fit(x_train, y_train)
y_pred = ppn.predict(x_test)
print('Incorrectly classified samples: %d' %(y_test != y_pred).sum())
print('Accuracy: %f '%accuracy_score(y_test.ravel(), y_pred))

from sklearn.metrics import confusion_matrix

confusion_matrix = confusion_matrix(y_test.ravel(), y_pred)
print(confusion_matrix)
sns.heatmap(confusion_matrix)
```

```
Incorrectly classified samples: 721
Accuracy: 0.677116
[[685 520]
 [201 827]]
```

```
: <matplotlib.axes._subplots.AxesSubplot at 0x27ea4e97488>
```



Rozdział 3

Podsumowanie

Dzięki metodzie MLP przy odpowiednim dobraniu parametrów można uzyskać dopasowanie na poziemie blizko 80%, co jest zadowalającym wynikiem, jednak bardzo blisko tego rozwiązania klasyfikuje się metoda regresji logistycznej. W celu poprawy wyniku sieci neuronowej możnaby spróbować zastosować metodę Gridsearch, która automatycznie dobierze parametry.

Dodatek A

Kod programu

```
import seaborn as sns
import numpy as np
import sklearn
import pickle
from sklearn.preprocessing import LabelEncoder
from sklearn import preprocessing
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import Perceptron
from sklearn.neural_network import MLPClassifier


df = pd.read_csv('bank_1.csv')
df.head().T

df.shape
df.describe()
df.info()
from matplotlib import pyplot as plt
sns.countplot(x=df['deposit'])
plt.title('Distribution of classes')
plt.xlabel('Target class')
df.deposit.value_counts()

sns.countplot(y='age', data=df)
for x in range(95, 101 , 1):
    print("{}% of people having age are less than equal to {}".format(x, df.age.quan
```

```
 lst = [df]
for column in lst:
    column.loc[column["age"] < 30,  'age_group'] = 30
    column.loc[(column["age"] >= 30) & (column["age"] <= 44), 'age_group'] = 40
    column.loc[(column["age"] >= 45) & (column["age"] <= 59), 'age_group'] = 50
    column.loc[column["age"] >= 60, 'age_group'] = 60


sns.countplot(x='age_group', data=df, hue='deposit')
sns.countplot(y='job', data=df)
sns.countplot(y='job', data=df, hue='deposit')
sns.countplot(y='marital', data=df)
sns.countplot(y='marital', data=df, hue='deposit')
df.education.value_counts()
sns.countplot(y='education', data=df)
sns.countplot(y='education', data=df, hue='deposit')
sns.countplot(x='pdays', data=df, hue='deposit')
df.pdays.value_counts()
df.poutcome.value_counts()
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sb

sb.pairplot(df, hue='deposit',diag_kind="kde")
df.drop('poutcome', axis=1, inplace=True)
df.drop('pdays', axis=1, inplace=True)
sb.pairplot(df, hue='deposit',diag_kind="kde")
le = preprocessing.LabelEncoder()
df.job = le.fit_transform(df.job)
df.marital= le.fit_transform(df.marital)
df.education = le.fit_transform(df.education)
df.deposit = le.fit_transform(df.deposit)

df.housing = le.fit_transform(df.housing)
df.loan = le.fit_transform(df.loan)

df.default = le.fit_transform(df.default)
df.contact = le.fit_transform(df.contact)
df.month = le.fit_transform(df.month)

df.head()
x = df.iloc[:,0:14]
y = df.iloc[:,14]
```

```
print(y)
x_train, x_test, y_train, y_test = model_selection.train_test_split(x, y, test_size=
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)


logistic_regression = LogisticRegression()
logistic_regression.fit(x_train, y_train)

prediction=logistic_regression.predict(x_test)
from sklearn.metrics import accuracy_score
accuracy_score(y_test, prediction)

prediction[0:50]
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test, prediction)
print(confusion_matrix)
import seaborn as sns;
sns.heatmap(confusion_matrix)

ppn = Perceptron(max_iter = 77, verbose=0, penalty="elasticnet", random_state = 1, t
ppn.fit(x_train, y_train)
y_pred = ppn.predict(x_test)
print('Incorrectly classified samples: %d' %(y_test != y_pred).sum())
print('Accuracy: %f '%accuracy_score(y_test.ravel(), y_pred))

from sklearn.metrics import confusion_matrix

confusion_matrix = confusion_matrix(y_test.ravel(), y_pred)
print(confusion_matrix)
sns.heatmap(confusion_matrix)


neural_network = MLPClassifier(hidden_layer_sizes=(200,100,50), activation = 'relu',
                               solver = 'adam', random_state=1, verbose = False, lea
neural_network.fit(x_train, y_train.ravel())

pr = neural_network.predict(x_test)
from sklearn.metrics import confusion_matrix
conf_matrix_neural_network = confusion_matrix(y_test.ravel(),pr)
print("Confusion_matrix:")
print(conf_matrix_neural_network)
```

```
import seaborn as sns;
sns.heatmap(conf_matrix_neural_network)
acc = accuracy_score(y_test.ravel(),pr)

print("Neural network model accuracy is {0:0.2f}".format(acc))

neural_network = MLPClassifier(hidden_layer_sizes=(40,40,40), activation = 'logistic
                               solver = 'adam',  random_state=1, verbose = False)
neural_network.fit(x_train, y_train.ravel())

pr = neural_network.predict(x_test)
from sklearn.metrics import confusion_matrix
conf_matrix_neural_network = confusion_matrix(y_test.ravel(),pr)
print("Confusion_matrix:")
print(conf_matrix_neural_network)
import seaborn as sns;
sns.heatmap(conf_matrix_neural_network)
acc = accuracy_score(y_test.ravel(),pr)

print("Neural network model accuracy is {0:0.2f}".format(acc))
```