



**ZAKŁAD SYSTEMÓW ZŁOŻONYCH**  
Wydział Elektrotechniki i Informatyki  
ul. Wincentego Pola 2, 35-959 Rzeszów, tel. 17 865 1340  
zsz.prz.edu.pl



**WYDZIAŁ  
ELEKTROTECHNIKI  
I INFORMATYKI**  
POLITECHNIKI RZESZOWSKIEJ

# Projektowanie systemów i sieci komputerowych – projekt

Predictive maintenance

Wykonali:  
Arkadiusz Połec, Jakub Marć



<b><i>Cel projektu</i></b>	<b>3</b>
<b><i>Wstęp</i></b>	<b>3</b>
Dane uczące	3
Język programowania oraz IDE	6
Przygotowanie danych	7
Algorytm klasyfikatora	11
<b><i>Przeprowadzone badania</i></b>	<b>14</b>
Predykcja wskazująca na możliwość wystąpienia awarii w ciągu najbliższych trzech dni	14
Predykcja wskazująca na możliwość wystąpienia awarii w ciągu najbliższych siedmiu dni	18
Predykcja wskazująca na możliwość wystąpienia awarii w ciągu najbliższych czternastu dni	22
<b><i>Wnioski</i></b>	<b>25</b>
<b><i>Linki</i></b>	<b>25</b>



## 1. Cel projektu

Głównym założeniem projektu było stworzenie systemu wczesnego ostrzegania o możliwości wystąpienia awarii urządzeń pamięci masowych, w oparciu o mechanizmy uczenia maszynowego. Obecnie ludzkość wytwarza i przechowuje ogromne ilości danych. Jednym ze sposobów zapewnienia względnego bezpieczeństwa dysku, jest przewidzenie możliwej awarii dysku, na którym owe dane się znajdują i uprzednio wykonanie jego kopii zapasowej. Tutaj nasz projekt ma swoje odzwierciedlenie, jako iż na podstawie odpowiednich parametrów S.M.A.R.T. (parametry systemu monitorowania i powiadamiania o błędach działania twardego dysku) będzie on w stanie określić możliwość wystąpienia awarii. Czas predykcji to 3, 7 oraz 14 dni, w czasie których to dysk może ulec awarii.



## 2. Wstęp

### 1.1. Dane uczące

Do nauki oraz testowania naszego systemu ekspertowego potrzebowaliśmy danych zawierających parametry S.M.A.R.T., czyli danych diagnostycznych, które obrazują stan dysku podczas jego pracy. Takiego typu dane udostępnia firma Backblaze.



Z ich strony można bezpłatnie pobrać dziesiątki gigabajtów danych dotyczących pracy tysięcy dysków twardych na przestrzeni czasu. W naszych badaniach postawiliśmy na ostatni kwartał roku 2021, w którym wybraliśmy 11 awarii dysków. Wyodrębniliśmy więc dane od początku października 2021 do dnia wystąpienia awarii. Wybrane dyski to:

Dysk	Model	Numer seryjny	Pojemność w B	Ostatni dzień przed awarią
	CT250MX500SSD1	2112E58C64F6	250059350016	2021-12-31
	TOSHIBA MG07ACA14TA	40X0A0P4F97G	14000519643136	2021-12-31



	TOSHIBA MG07ACA14TA	78B0A030F97G	14000519643136	2021-12-31
	ST4000DM000	Z304GSQ0	4000787030016	2021-12-31
	ST4000DM000	Z305AQQP	4000787030016	2021-12-30
	ST8000DM002	ZA11X9R8	8001563222016	2021-12-31
	ST8000DM002	ZA12KZ0K	8001563222016	2021-12-31
	ST8000NM0055	ZA14EE4H	8001563222016	2021-12-30
	ST8000NM0055	ZA16DQA0	8001563222016	2021-12-31
	ST8000NM0055	ZA16DXEY	8001563222016	2021-12-31
	ST12000NM0007	ZJV4370N	12000138625024	2021-12-30

Dane mieszczą się w pliku .csv i zawierają podstawowe parametry takie jak:

`_id` - id pola z bazy danych MongoDB,

`date` - data pomiaru,

`serial_number` - numer seryjny dysku,

`model` - model dysku,

`capacity_bytes` - pojemność dysku w bajtach,

`failure` - parametr określający dwa stany: 0 – dysk w tym dniu pracował normalnie, 1 – dysk w tym dniu uległ awarii.

Oprócz powyższych parametrów dane zawierają również szereg cech S.M.A.R.T w postaci RAW(dane surowe).

Poniżej można znaleźć typowe znaczenie surowych wartości S.M.A.R.T. używanych w projekcie.



Znormalizowane wartości są zwykle mapowane, aby wyższe wartości były bardziej miarodajne, natomiast wyższe wartości atrybutów surowych mogą różnić się w zależności od atrybutu i producenta! Na przykład, znormalizowana wartość atrybutu S.M.A.R.T 5 (0x05) „Liczba ponownie relokowanych sektorów” może zmniejszać się wraz ze wzrostem liczby ponownie przydzielonych sektorów, podczas gdy surowa wartość atrybutu często wskazuje na rzeczywistą liczbę sektorów, które zostały ponownie przydzielone. Dostawcy i producenci nie są w żaden sposób zobowiązani do przestrzegania tych konwencji!

Ponieważ nie ma jednoznacznej zgodności, ani standardu co do dokładnych definicji atrybutów i jednostek miary, listę można uznać jedynie jako wskazówkę lub przewodnik:

**S.M.A.R.T 1 (0x01) - Read Error Rate** - Oznacza wskaźnik odczytu błędów występujących podczas odczytywania danych z powierzchni dysku. Wartość, która nie równa się zero może oznaczać problemy z powierzchnią dysku, głowicami odczytu/zapisu lub z niezbyt dokładnie umieszczonymi głowicami na ścieżce zapisu. Napędy Seagate często wskazują na nieprzetworzoną wartość, która jest wysoka w nowych napędach, która jednak nie oznacza uszkodzenia.

**S.M.A.R.T 2 (0x02)- Throughput Performance** - Ogólna przepustowość twardego dysku. Jeśli wartość atrybutu maleje to prawdopodobnie zbliżają się problemy z dyskiem.

**S.M.A.R.T 3 (0x03)- Spinup time** - Czas do rozkręcenia talerzy dysku oznacza czas w ms w jakim rozpędzają się talerze dysku (od 0 RPM do pełnej prędkości).

**S.M.A.R.T 4 (0x04)- Start/Stop count** - Ilość cykli start/stop – talerzy dysku. Cykl rozpoczyna się przy uruchomieniu dysku, lub przy wyjściu z trybu uśpienia.

**S.M.A.R.T 5 (0x05)- Reallocated Sectors Count** - Liczba ponownie realokowanych (na nowo przydzielonych) sektorów. Gdy twardy dysk znajduje błąd zapisu lub odczytu, zaznacza sektor do ponownego przydzielenia i przesyła dane do specjalnie przeznaczonego obszaru (zapasowego). Proces znany też jako "remapowanie". Z tego powodu w nowych dyskach nie można odnaleźć "bad sektorów" ze względu na fakt, że nie są one wykrywane podczas sprawdzania powierzchni dysku. Wszystkie "bad sektory" są ukryte poprzez zastąpienie ich nowymi, z zapasowego obszaru. Gdy liczba ponownie przydzielonych sektorów wzrasta to prędkość odczytu i zapisu maleje (głowica dysku musi przesunąć się ponad obszar w którym są przechowywane zapasowe sektory). Nieprzetworzona wartość zwykle oznacza liczbę złych sektorów, które zostały znalezione. Im większa wartość, tym więcej sektorów ulega ponownemu przydzieleniu, co niekorzystnie wpływa na pracę dysku.

**S.M.A.R.T 7 (0x07)- Seek Error Rate** - Liczba błędów pozycjonowania – częstotliwość występowania błędów podczas wyszukiwania przez głowice. Każdy dysk twardy składa się z pozycjonera głowicy, jeżeli system ten jest uszkodzony to liczba błędów bardzo szybko wzrasta.

**S.M.A.R.T 8 (0x08)- Seek timer performance** - Wydajność podczas wyszukiwania - gdy atrybut maleje to oznacza, że zaistniał problem z podsystemem mechanicznym dysku.

**S.M.A.R.T 9 (0x09)- Power-On Hours** - Czas pracy dysku - informuje o ilości godzin, które przepracował dysk twardy.

**S.M.A.R.T 10 (0x0A)- Spin Retry Count (Spin-up retries)** - Liczba prób rozpędzenia talerzy dysku. Atrybut ten przechowuje całkowitą liczbę prób rozpoczęcia rozpędzenia talerzy dysku (do osiągnięcia pełnej prędkości obrotowej). Wzrost wartości atrybutu oznacza problemy z podsystemem mechanicznym dysku.



S.M.A.R.T 11 (0x0B)- Calibration retry count - Liczba ponownych rekaliibracji - pokazuje całkowitą liczbę żądań rekaliibracji, jeżeli pierwsza próba rekaliibracji nie powiodła się.

S.M.A.R.T 12 (0x0C)- Power cycle count - Liczba cykli zasilania- Atrybut oznacza sumę pełnych cykli zasilania dysku.

S.M.A.R.T 13 (0x0D)- Soft read error rate - Ilość poprawionych błędów odczytu zgłoszonych do systemu operacyjnego.

S.M.A.R.T 15 (0x0F)- User Capacity - Pojemność przeznaczona dla użytkownika.

S.M.A.R.T 16 (0x10)- Spare Blocks Available - Dostępne bloki zapasowe.

S.M.A.R.T 17 (0x11)- Remaining Spare Blocks - Pozostałe bloki zapasowe.

S.M.A.R.T 220 (0xDC)- Disk Shift - Przesunięcie talerzy od osi. Przesunięcie najczęściej jest wynikiem upadku/uderzenia dysku.

S.M.A.R.T 222 (0xDE)- Loaded Hours - Łączny czas pracy akuratora głowic magnetycznych podczas normalnego działania.

S.M.A.R.T 223 (0xDF)- Load/unload retry count - Łączny czas pracy akuratora głowic magnetycznych podczas operacji odczytu/zapisu/pozycjonowania.

S.M.A.R.T 224 (0xE0)- Load Friction - Łączny czas użycia akuratora głowic spowodowany oporem na skutek tarcia części mechanicznych dysku podczas pracy.

S.M.A.R.T 242 (0xF2)- Total LBA-s Read - Całkowita liczba odczytanych sektorów (dotyczy dysków WD).

S.M.A.R.T 250 (0xFA)- Read Error Retry Rate - Liczba błędów odczytu z dysku.

S.M.A.R.T 251 (0xFB)- Minimum Spares Remaining - pozostała minimalna ilość bloków zamiennych - wskazuje liczbę pozostałych zapasowych bloków jako procent całkowitej liczby dostępnych zapasowych bloków.

S.M.A.R.T 254 (0xFE)- Free Fall Protection - Liczba wykrytych przez czujnik przyśpieszenia przeciążeń

## 1.2. Język programowania oraz IDE



W projekcie wykorzystano język programowania **Python** jako główny język programistyczny skryptu do przygotowania danych i algorytmu uczenia maszynowego, gdyż udostępnia wiele przydatnych bibliotek do algorytmów sztucznej inteligencji i wyświetlania wyników.

Środowisko programistyczne wybrane do tworzenia skryptu to Jupyter Notebook.



**Jupyter Notebook** (wcześniej IPython Notebooks) to REPL (proste, interaktywne środowisko programowania) oparte na przeglądarce, zawierające uporządkowaną listę komórek wejścia/wyjścia, które mogą zawierać kod, tekst, wyrażenia matematyczne, wykresy i multimedia. Notatnik pod warstwą interfejsu jest dokumentem JSON zgodnym ze schematem, zwykle kończącym się rozszerzeniem „.ipynb”.

Instrukcję instalacji można znaleźć pod adresem witryny dewelopera Jupyter: <https://jupyter.org/install>

### 1.3. Przygotowanie danych

W celu wczytania wszystkich ramek danych należy utworzyć listę z nazwami numerów seryjnych, według których są nazywane pliki ramek:

```
collections = [  
    '2112E58C64F6', #dysk używany tylko do pred. 7 i 3 dniowej.  
    '40X0A0P4F97G',  
    '78B0A030F97G',  
    'Z304GSQ0',  
    'Z305AQQP',  
    'ZA11X9R8',  
    'ZA12KZ0K',  
    'ZA14EE4H',  
    'ZA16DQA0',  
    'ZA16DXEY',  
    'ZJV4370N'  
]
```

Następnie należy utworzyć ramkę danych do której wczytamy dane z plików. W tym celu można użyć biblioteki pandas, służącej do manipulacji i analizy danych.

```
import pandas as pd  
X = pd.DataFrame()  
y = pd.DataFrame()
```

Po stworzeniu DataFrame, w pętli wczytywane i przetwarzane są dane z wielu plików. Wczytać dane możemy na 2 sposoby:

W przypadku plików lokalnych należy wykorzystać metodę `pd.read_csv()` z biblioteki pandas, podając odpowiednią ścieżkę do pliku:

```
for col in collections:  
    # Get a single dataframe from local file  
    cdf = pd.read_csv('all_pred_14/' + col + '_raw.csv')
```

W przypadku wczytywania plików z bazy danych MongoDB można utworzyć odpowiednią funkcję, która będzie odpowiedzialna za wczytywanie. Należy w tym przypadku zaimportować pakiet MongoClient:



```
from pymongo import MongoClient

def getDFfromMongo(collection):

    client = MongoClient(
        'URI do połączenia z klientem MongoDB')
    filter = {}
    project = {}

    result = client['all_pred_14'][collection].find(
        filter=filter,
        projection=project
    )

    df = pd.DataFrame(list(result))
    return df
```

Przy wywołaniu metody `find` należy podać nazwę kolekcji z bazy danych, którą używamy do wczytania (w tym przypadku użyto kolekcji `'all_pred_14'`).

Następnie można wczytać ramkę danych:

```
for col in collections:
    # Get a single dataframe from MongoDB
    cDF = getDFfromMongo(col)
```

Po wczytaniu dane można przetworzyć (przetwarzamy dane w dalszym ciągu w pętli `for`). Usuwamy niepotrzebne kolumny za pomocą metody `pandas.DataFrame.drop`. Podany argument `axis='columns'` oznacza że usuwamy kolumny.

```
# Dropping unnecessary columns
cDF = cDF.drop(['_id', 'date', 'serial_number', 'model',
               'capacity_bytes'], axis='columns')
```

Następnie przypisujemy odpowiednie kolumny ramki do zmiennej funkcji `cX` i zmiennej odpowiedzi `cY`:

```
# Putting feature variable to X
cX = cDF.drop('failure', axis='columns')

# Putting response variable to y
cy = cDF['failure']
```

W kolejnym etapie usuwamy wszystkie kolumny, których wszystkie wiersze są puste. Używamy do tego metody `pandas.DataFrame.dropna`, która usuwa wszystkie kolumny z pustymi wartościami. Podajemy argument `how='all'`, który określa że usuwamy tylko kolumny, w których wszystkie wiersze zawierają puste wartości.

```
# Dropping empty columns
cX = cX.dropna(axis='columns', how='all')
```





Następnie usuwamy wszystkie kolumny, w których wartości wierszy się nie zmieniają. W tym celu unikniemy danych, które będą zbędne przy uczeniu maszynowym. Używamy do tego listy składanej oraz metody `pandas.DataFrame.nunique`, która zwraca liczbę unikatowych elementów w danej kolumnie.

```
# Dropping columns with the same values
cX = cX[[column for column in cX if cX[column].nunique()>1]]
```

Na końcu pętli `for` łączymy ramki pojedynczego dysku z główną ramką, która będzie przechowywać dane wszystkie z dysków za pomocą metody `pandas.concat`:

```
# Join all dataframes
X = pd.concat([X, cX])
y = pd.concat([y, cy])
```

Po wyświetleniu ramki `X` można zauważyć, że po połączeniu danych z różnych dysków, które po przetworzeniu zawierały różne S.M.A.R.T.y powstały puste wartości:

```
In [28]: X
Out[28]:
```

	smart_18_raw	smart_222_raw	smart_17_raw	smart_1_raw	smart_2_raw	smart_9_raw	smart_12_raw	smart_13_raw	smart_163_raw	smart_250_raw	smart_...
0	1	2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1	5	2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2	14	2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
3	18	2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
4	26	3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	...	
84	19832	NaN	78	83	211167448	NaN	89	859491093	NaN	NaN	
85	19864	NaN	78	83	183244312	NaN	89	866898570	NaN	NaN	
86	19896	NaN	78	82	170677416	NaN	89	871066612	NaN	NaN	
87	19911	NaN	78	80	108041344	NaN	89	872948153	NaN	NaN	
88	19943	NaN	78	77	46762864	NaN	89	877616667	NaN	NaN	

926 rows × 14 columns

Usuujemy je za pomocą metody `pandas.DataFrame.fillna`, która wypełnia wszystkie puste wartości(`NaN`) podaną wartością. Używamy wartości `0`.

```
# Convert NaN values to zero values
X = X.fillna(0)
```

```
In [30]: X
Out[30]:
```

	smart_18_raw	smart_222_raw	smart_17_raw	smart_1_raw	smart_2_raw	smart_9_raw	smart_12_raw	smart_13_raw	smart_163_raw	smart_250_raw	smart_...
0	1	2	0	0	0	0	0	0	0	0	
1	5	2	0	0	0	0	0	0	0	0	
2	14	2	0	0	0	0	0	0	0	0	
3	18	2	0	0	0	0	0	0	0	0	
4	26	3	0	0	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	
84	19832	0	78	83	211167448	0	89	859491093	0	0	
85	19864	0	78	83	183244312	0	89	866898570	0	0	
86	19896	0	78	82	170677416	0	89	871066612	0	0	
87	19911	0	78	80	108041344	0	89	872948153	0	0	
88	19943	0	78	77	46762864	0	89	877616667	0	0	

926 rows × 14 columns



Ze względu na dużą rozpiętość wartości, dane można znormalizować. W tym przypadku należy zaimportować MinMaxScaler z biblioteki sklearn. Do normalizacji używamy estymatora sklearn.preprocessing.MinMaxScaler oraz metody fit\_transform estymatora MinMaxScaler. Argument przy tworzeniu estymatora feature\_range=(-1, 1) oznacza zakres do jakiego chcemy znormalizować dane.

```
from sklearn.preprocessing import MinMaxScaler
# Normalize the data
sc = MinMaxScaler(feature_range=(-1, 1))
normed_X = pd.DataFrame(sc.fit_transform(X), columns=X.columns)
```

Wynik normalizacji:

```
In [31]: normed_X
```

```
Out[31]:
```

	smart_18_raw	smart_222_raw	smart_17_raw	smart_1_raw	smart_2_raw	smart_9_raw	smart_12_raw	smart_13_raw	smart_163_raw	smart_250_raw	smart
0	-1.000000	-0.333333	-1.00	-1.000000	-1.000000	-1.0	-1.000000	-1.000000	-1.0	-1.0	
1	-0.999849	-0.333333	-1.00	-1.000000	-1.000000	-1.0	-1.000000	-1.000000	-1.0	-1.0	
2	-0.999510	-0.333333	-1.00	-1.000000	-1.000000	-1.0	-1.000000	-1.000000	-1.0	-1.0	
3	-0.999360	-0.333333	-1.00	-1.000000	-1.000000	-1.0	-1.000000	-1.000000	-1.0	-1.0	
4	-0.999059	0.000000	-1.00	-1.000000	-1.000000	-1.0	-1.000000	-1.000000	-1.0	-1.0	
...	...	...	...	...	...	...	...	...	...	...	
921	-0.253182	-1.000000	0.95	0.383333	0.731541	-1.0	0.893617	-0.595313	-1.0	-1.0	
922	-0.251977	-1.000000	0.95	0.383333	0.502576	-1.0	0.893617	-0.591825	-1.0	-1.0	
923	-0.250772	-1.000000	0.95	0.366667	0.399529	-1.0	0.893617	-0.589862	-1.0	-1.0	
924	-0.250207	-1.000000	0.95	0.333333	-0.114077	-1.0	0.893617	-0.588977	-1.0	-1.0	
925	-0.249002	-1.000000	0.95	0.283333	-0.616552	-1.0	0.893617	-0.586778	-1.0	-1.0	

926 rows × 14 columns

W ten sposób dane zostały przygotowane do uczenia maszynowego.



## 1.4. Algorytm klasyfikatora

Jako klasyfikator wybraliśmy algorytm RandomForest, który posiada swoją implementację w Pythonie. Polega on na tworzeniu wielu drzew decyzyjnych z losowych parametrów oraz późniejszym podejmowaniu końcowej decyzji na podstawie większości wyników tych drzew składowych. Algorytm ten cechuje się dużą skutecznością klasyfikacji oraz łatwością implementacji, stąd wybór padł właśnie na niego.

Importujemy do projektu klasyfikator oraz model selekcji:

```
from sklearn.ensemble import RandomForestClassifier  
from sklearn.model_selection import train_test_split
```

Będziemy również potrzebować biblioteki numpy do spłaszczenia ramki zmiennej odpowiedzi (y):

```
import numpy as np
```

Dzielimy przygotowane dane na dane uczące oraz dane testowe. Używamy do tego modelu selekcji `train_test_split`, który dzieli tablice lub macierze na losowe podzbiory uczące i testowe.

Jako pierwsze argumenty metody podajemy nasze dane:

- znormalizowaną zmienną funkcji (`normed_X`),

- zmienną odpowiedzi (y) spłaszczoną do tablicy jednowymiarowej za pomocą metody `numpy.ravel()`,

Jako kolejne argumenty podajemy pozostałe parametry:

- `test_size=.X` - rozmiar danych testowych (zalecane 20-40% całkowitych danych)

- `random_state=42` - steruje tasowaniem stosowanym do danych przed zastosowaniem podziału (zalecana jest standardowa wartość - 42)

```
# Splitting the data into train and test
```

```
training, testing, training_labels, testing_labels = train_test_split(  
    normed_X, y.values.ravel(), test_size=.3, random_state=42)
```

Następnie tworzymy klasyfikator:

```
clf = RandomForestClassifier()
```

Należy zauważyć że używamy domyślnych parametrów klasyfikatora. Jest to spowodowane, tym że z domyślnymi parametrami osiągnęliśmy satysfakcjonujące rezultaty, a zmiany parametrów nie wpływały znacząco na wynik uczenia.

Następnie umieszczamy dane w klasyfikatorze:

```
clf.fit(training, training_labels)
```

Uruchamiamy algorytm przewidywania. Jako argument podajemy dane testowe.

```
preds = clf.predict(testing)
```

Następnie możemy wyświetlić wynik przewidywania dla danych uczących i danych testowych:

```
print(clf.score(training, training_labels))  
print(clf.score(testing, testing_labels))
```



```
In [35]: # Make Classifier
clf = RandomForestClassifier()
clf.fit(training, training_labels)
preds = clf.predict(testing)
print(clf.score(training, training_labels))
print(clf.score(testing, testing_labels))

1.0
0.9820143884892086
```

Wynik przewidywania danych uczących powinien wynosić 100%. W innym przypadku należy zmienić parametry klasyfikatora.

Dla wygodnego przedstawienia wyniku można użyć tablicy pomyłek, którą oferuje biblioteka `sklearn`, biblioteki `seaborn` która pozwala na wyświetlenie tablicy kodowanej kolorami oraz biblioteki `matplotlib.pyplot` do wyświetlania wykresów.

```
from sklearn.metrics import confusion_matrix
import seaborn as sn
import matplotlib.pyplot as plt
```

Tworzenie wykresu:

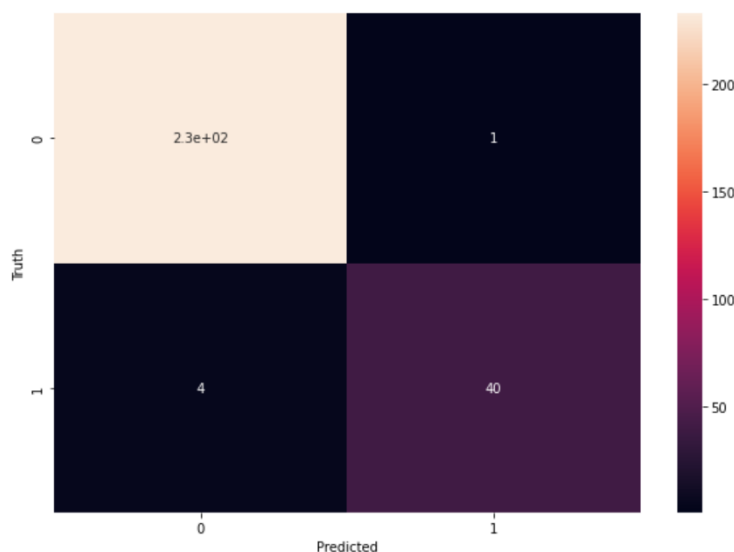
```
%matplotlib inline
plt.figure(figsize=(10, 7))
sn.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
pd.DataFrame(clf.feature_importances_, index=training.columns).sort_values(
    by=0, ascending=False)
```

W ten sposób można stworzyć poniższy wykres kodowany kolorami:

```
In [36]: cm = confusion_matrix(testing_labels, preds)

%matplotlib inline
plt.figure(figsize=(10, 7))
sn.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
Out[36]: Text(69.0, 0.5, 'Truth')
```





Odczytywanie tablicy pomyłek:

		Klasa predykowana	
		pozytywna	negatywna
Klasa rzeczywista	pozytywna	prawdziwie  pozytywna (TP)	falszywie  pozytywna (FP)
	negatywna	falszywie  negatywna (FN)	prawdziwie  negatywna (TN)

Klasyfikator RandomForest umożliwia także sprawdzenie w jakim stopniu każdy z atrybutów wpłynął na wynik nauczania modelu. Dane te są przechowywane w atrybucie `feature_importances_` klasyfikatora `RandomForestClassifier()`.

Można te dane wyświetlić jako ramka danych:

```
pd.DataFrame(clf.feature_importances_, index=training.columns).sort_values(  
    by=0, ascending=False)
```

Out[14]:

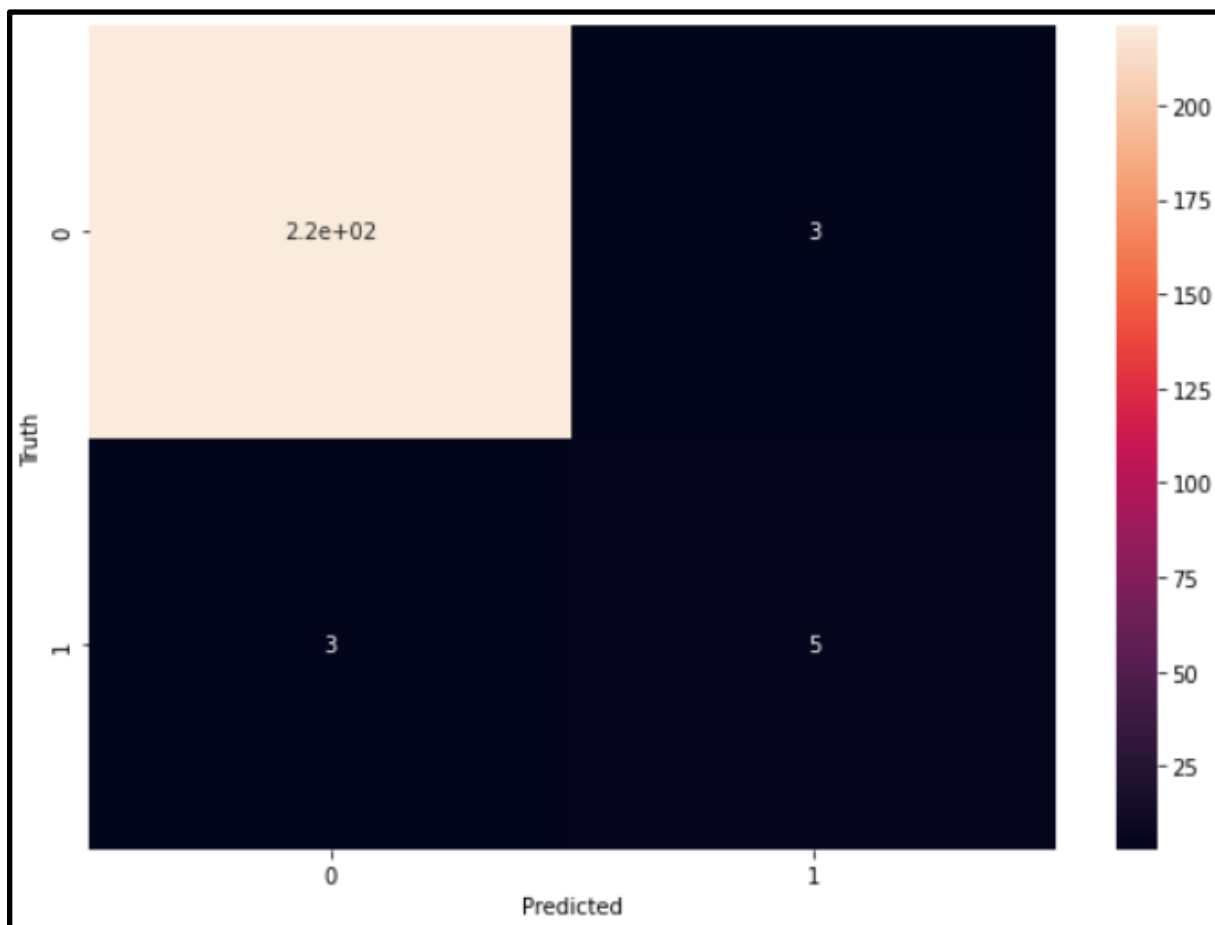
	0
smart_18_raw	0.268818
smart_13_raw	0.199842
smart_17_raw	0.180918
smart_11_raw	0.108598
smart_255_raw	0.067436
smart_2_raw	0.035815
smart_1_raw	0.031921
smart_12_raw	0.029266
smart_222_raw	0.021649
smart_9_raw	0.021307
smart_163_raw	0.015608
smart_10_raw	0.015015
smart_251_raw	0.002284
smart_250_raw	0.001522



### 3. Przeprowadzone badania

#### 3.1. Predykcja wskazująca na możliwość wystąpienia awarii w ciągu najbliższych trzech dni

a) dla stosunku danych uczących i testujących 3:1



Stopień 'ważności' danej cechy przy podejmowaniu decyzji



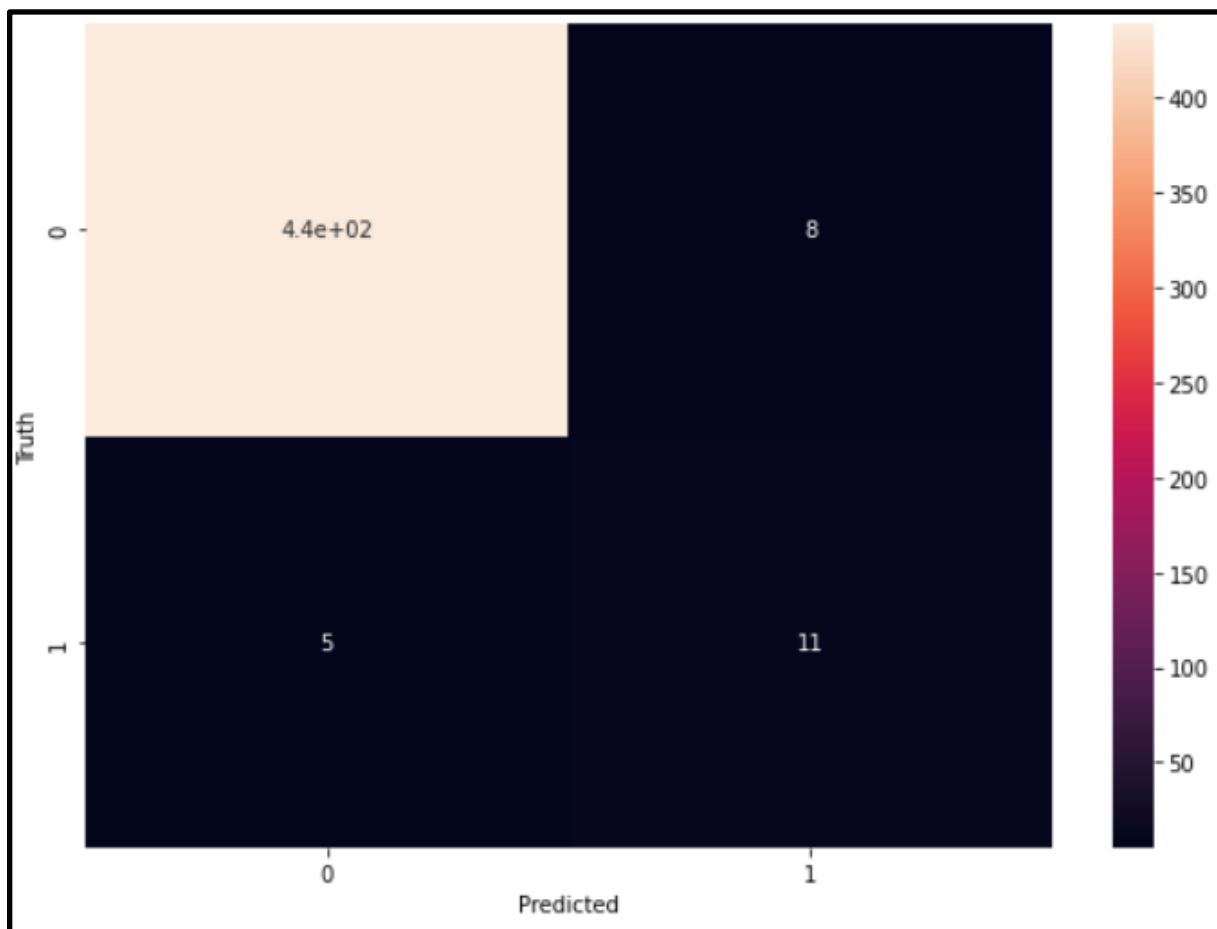
smart_18_raw	0.312094
smart_13_raw	0.177541
smart_17_raw	0.119182
smart_11_raw	0.083877
smart_222_raw	0.057822
smart_255_raw	0.057403
smart_2_raw	0.052822
smart_12_raw	0.042325
smart_1_raw	0.032938
smart_251_raw	0.030485
smart_9_raw	0.014717
smart_10_raw	0.008968
smart_163_raw	0.007533
smart_250_raw	0.002293

Skuteczność predykcji dla danych testowych

0.9741379310344828



b) dla stosunku danych uczących i testujących 1:1







Stopień 'ważności' danej cechy przy podejmowaniu decyzji

smart_18_raw	0.275465
smart_13_raw	0.145206
smart_17_raw	0.143995
smart_11_raw	0.125753
smart_255_raw	0.057242
smart_2_raw	0.051918
smart_1_raw	0.051081
smart_222_raw	0.034064
smart_251_raw	0.032393
smart_12_raw	0.023586
smart_9_raw	0.022658
smart_163_raw	0.019600
smart_10_raw	0.010244
smart_250_raw	0.006794

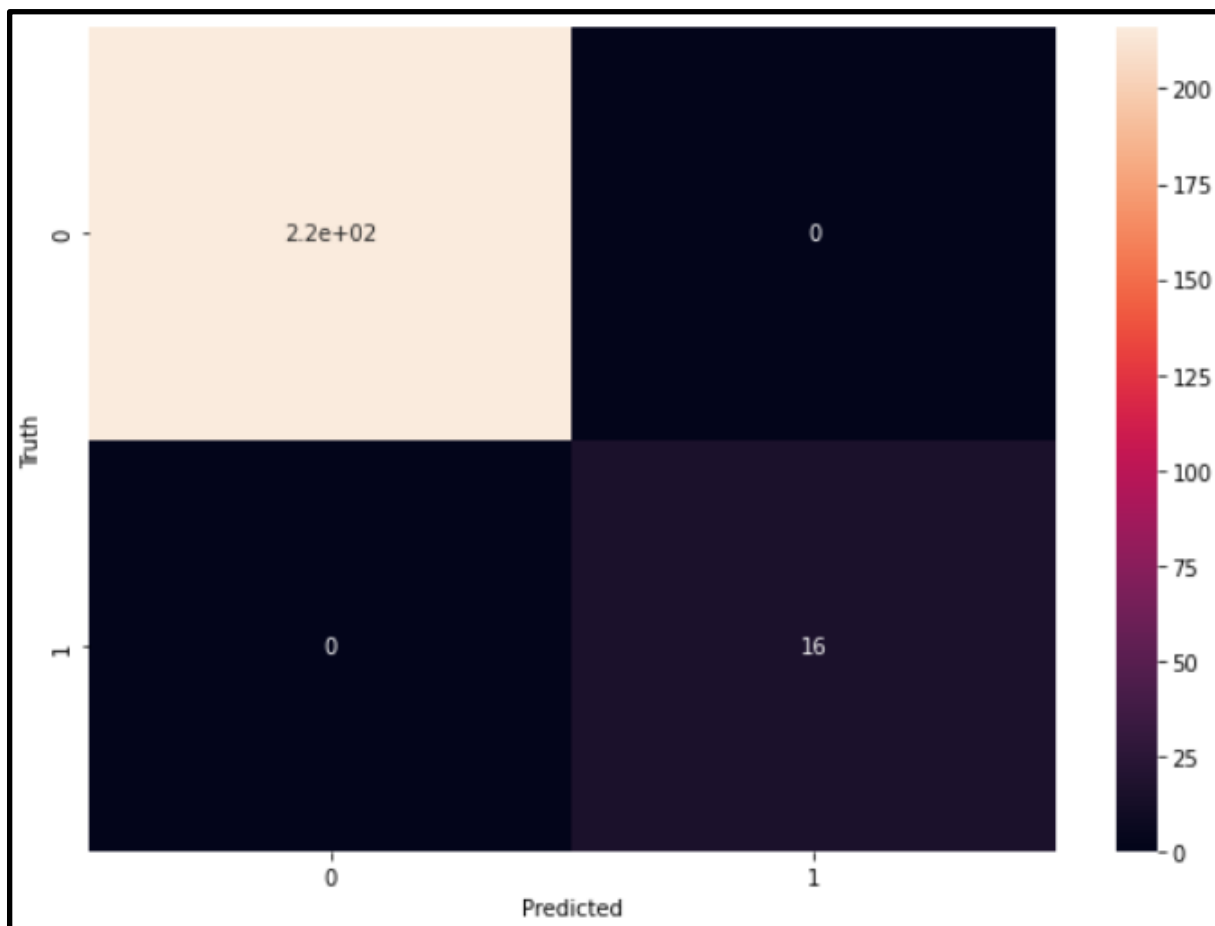
Skuteczność predykcji dla danych testowych

0.9676025917926566



### 3.2. Predykcja wskazująca na możliwość wystąpienia awarii w ciągu najbliższych siedmiu dni

a) dla stosunku danych uczących i testujących 3:1





Stopień 'ważności' danej cechy przy podejmowaniu decyzji

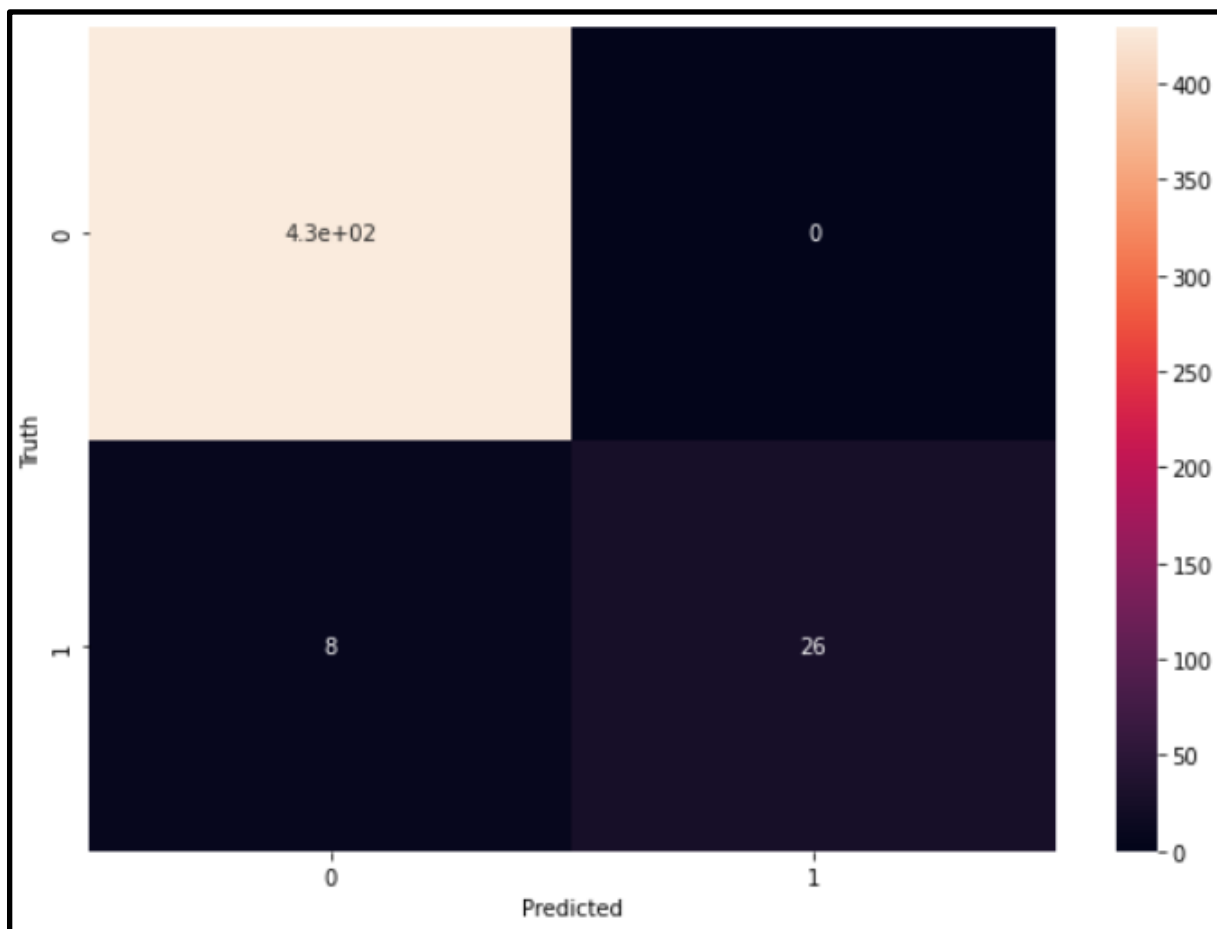
smart_18_raw	0.315237
smart_13_raw	0.165860
smart_17_raw	0.137121
smart_11_raw	0.108088
smart_255_raw	0.057070
smart_222_raw	0.051329
smart_1_raw	0.045856
smart_2_raw	0.042759
smart_12_raw	0.027572
smart_10_raw	0.014634
smart_163_raw	0.012599
smart_9_raw	0.009981
smart_251_raw	0.007281
smart_250_raw	0.004613

Skuteczność predykcji dla danych testowych

1.0



b) dla stosunku danych uczących i testujących 1:1





Stopień 'ważności' danej cechy przy podejmowaniu decyzji

smart_18_raw	0.262996
smart_13_raw	0.190035
smart_17_raw	0.147999
smart_11_raw	0.107215
smart_255_raw	0.067706
smart_2_raw	0.052928
smart_1_raw	0.049803
smart_222_raw	0.033935
smart_12_raw	0.032196
smart_163_raw	0.021500
smart_9_raw	0.017870
smart_10_raw	0.009270
smart_250_raw	0.006548
smart_251_raw	0.000000

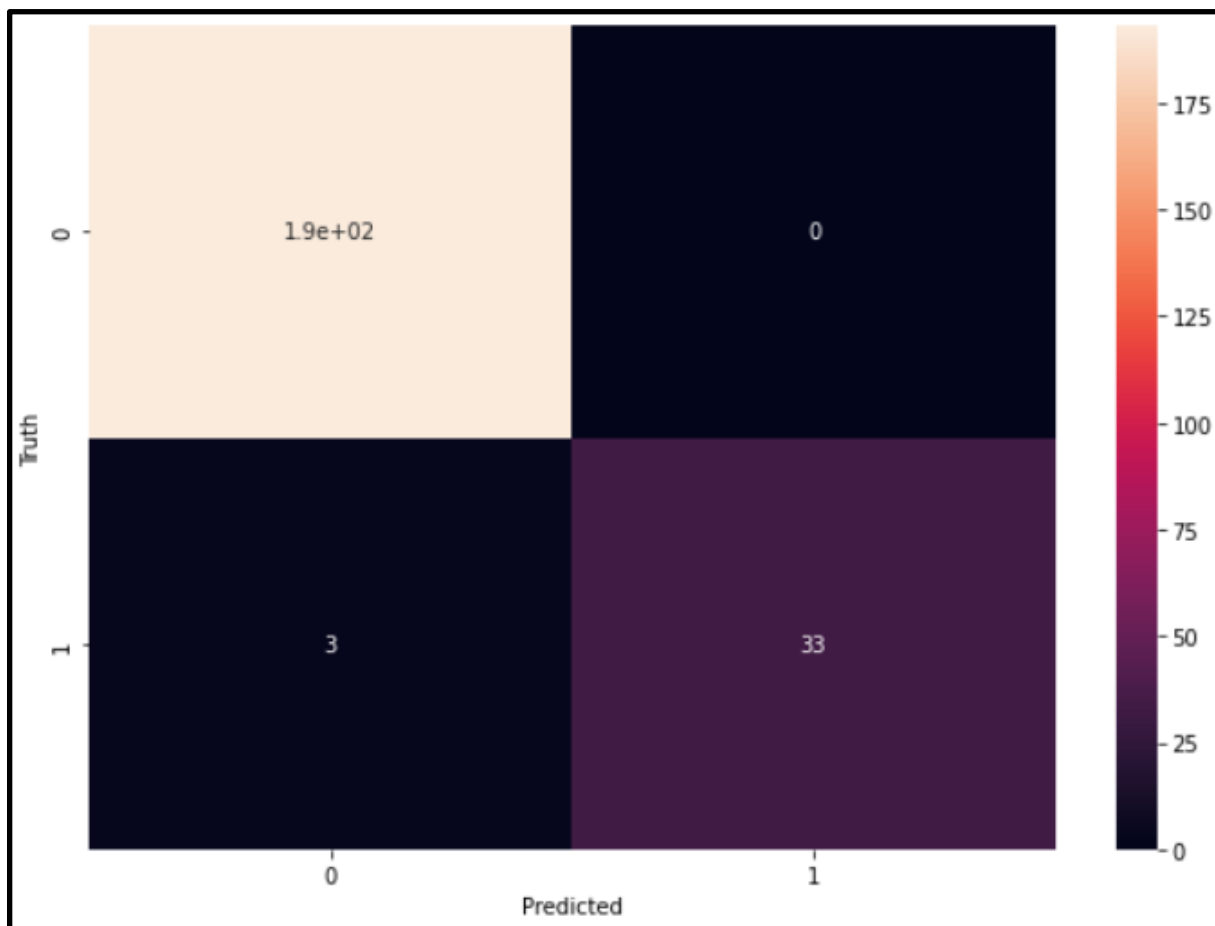
Skuteczność predykcji dla danych testowych

0.9848812095032398



### 3.3. Predykcja wskazująca na możliwość wystąpienia awarii w ciągu najbliższych czternastu dni

a) dla stosunku danych uczących i testujących 3:1





Stopień 'ważności' danej cechy przy podejmowaniu decyzji

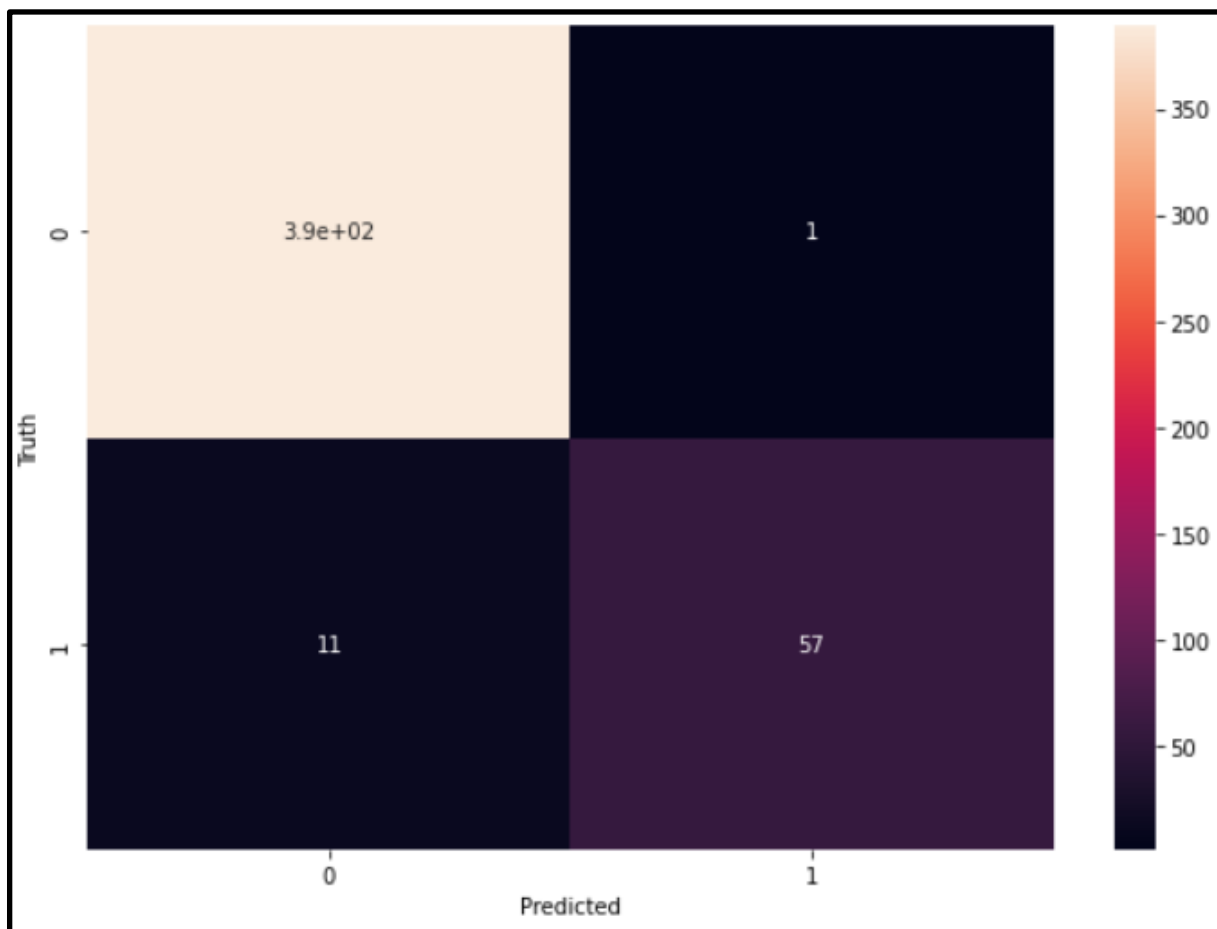
smart_18_raw	0.288789
smart_13_raw	0.200375
smart_17_raw	0.161905
smart_11_raw	0.131680
smart_255_raw	0.056495
smart_12_raw	0.044043
smart_10_raw	0.030074
smart_2_raw	0.027483
smart_1_raw	0.023464
smart_163_raw	0.016451
smart_9_raw	0.015917
smart_250_raw	0.003325
smart_251_raw	0.000000

Skuteczność predykcji dla danych testowych

0.9868995633187773



b) dla stosunku danych uczących i testujących 1:1







Stopień 'ważności' danej cechy przy podejmowaniu decyzji

smart_18_raw	0.242219
smart_13_raw	0.204955
smart_17_raw	0.164223
smart_11_raw	0.144024
smart_255_raw	0.072779
smart_12_raw	0.048930
smart_2_raw	0.037287
smart_1_raw	0.030532
smart_10_raw	0.024685
smart_9_raw	0.015594
smart_163_raw	0.012180
smart_250_raw	0.002590
smart_251_raw	0.000000

Skuteczność predykcji dla danych testowych

0.9759825327510917

## 4. Wnioski

Dla każdego z trzech wariantów 3, 7, 14 osiągnęliśmy przyzwoitą skuteczność predykcji (96,4%-100,0%), pomimo zastosowaniu różnych stosunków danych uczących i testujących. W większości przypadków wszystkie cechy miały swój udział przy podejmowaniu decyzji.

## 5. Linki

- [Dane udostępniane przez Backblaze](#)
- kod programu w postaci notebook'a pakietu Jupyter dla języka Python,
- dane uczące w postaci plików 'json' oraz 'csv'.