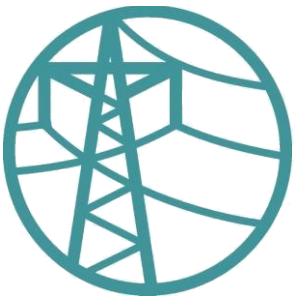


WYDZIAŁ
ELEKTROTECHNIKI
I INFORMATYKI
POLITECHNIKI RZESZOWSKIEJ

Damian Kobyliński

Analiza zużycia wybranego elementu w oparciu o metodę zubożania cech oraz rekordów

Zadanie Projektowe

Opiekun pracy:
Marek Bolanowski

Rzeszów, 2023

Spis treści

1. Wprowadzenie

W pozniższym sprawozdaniu projektowym chciałbym opisać zadanie projektowe oparte na analize zurzycia wybranego elementu w oparciu o metodę zuborzania cech jak i ilosci rekordów badanego elementu. Jako baza danych do analizy wykorzystałem dane z Instytucji Energii Naturalnej na 14-stu bateriach NMC-LCO 18650 z pojemnością 2.8 Ah. Baterie zostaly przetestowane na 1000 cyklach w temperaturze 25^{o}C.

Projekt wykonany głównie w celu zaznajomienia z podstawowymi funkcjami języka programowania Pythona. Dodatkowo zostały wykorzystane dodatkowe biblioteki:

- Pandas – biblioteka do manipulacji i analizy danych,
- Numpy - biblioteka do manipulacji i analizy danych,
- Seaborn – biblioteka pozwalajaca zwizualizować informacje o danej bazie danych,
- Matplotlib – biblioteka służąca do rysowania wykresów i wizualizacji danych,
- Sklearn – biblioteka zawierająca pakiety funkcji do uczenia maszynowego,
- Tensorflow – biblioteka służąca do obliczeń numerycznych i tworzenia modeli uczenia maszynowego,
- Keras – wysokopoziomowa biblioteka uczenia maszynowego,

2. Przedstawienie projektu

W tym rozdziale opisany został cały szkielet projektu.

2.1. Biblioteki

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix,accuracy_score, roc_curve

import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers.core import Dense, Activation, Dropout, Embedding
from keras import backend as K
from sklearn.metrics import f1_score
```

Zdjęcie 1: Przedstawienie wykorzystanych bibliotek

2.2. Baza danych

Wykorzystana baza danych:

```
battery_dataset = pd.read_csv("Battery_RUL.csv",sep=",",header=None)
df = battery_dataset.iloc[1:]
columns = ['Cycle_Index','dischargeTime','Decrement 3.6-3.4V (s)','Max. Voltage Dischar. (V)','Min. Voltage Charg. (
df.columns = columns
X = df.drop(['RUL','Cycle_Index','Change'],axis=1)
Y = df['Change']
X = X.astype(float)
Y = Y.astype(float)
Y = pd.DataFrame(Y)
```

Zdjęcie 2: Przedstawienie bazy danych

W przedstawionym powyżej kodzie, wyczytywany jest plik typu CSV. Tworzona jest kopia tego pliku z usunięciem pierwszego wiersza w którym zawarte są nazwy kolumn – je umieszczamy ręcznię. Następnie dzielona jest baza danych na kolumny z zawartymi cechami jak i z kolumnami labelów.

W trakcie zubażania zarówno kolumn z cechami, jak i ilości rekordów, zmienia się ilość danych.

- Zubażanie co 2 kolumne z cechami

```
X_2drop = df.drop(['RUL','Cycle_Index','Change','dischargeTime','Max. Voltage Dischar. (V)','Time at 4.15V (s)','Cha
X_2drop = X_2drop.astype(float)
X_train_2drop,X_test_2drop,Y_train_2drop,Y_test_2drop = train_test_split(X_2drop,Y, test_size=0.30)
```

- Zubażanie do jednej kolumny z cechami

```
X_1 = df.drop(['RUL','Cycle_Index','Change','Decrement 3.6-3.4V (s)','Max. Voltage Dischar. (V)','Min. Voltage Charg
X_1 = X_1.astype(float)
X_train_1,X_test_1,Y_train_1,Y_test_1 = train_test_split(X_1,Y, test_size=0.30)
last_number_of_nodes = len(X_train_1)
```

- Zubożanie co o połowe liczby rekordów w bazie przez usunięcię co 2 rekordu

```
by_2rows_drop = []
for i in range(1,len(X)):
    if i%2 == 0:
        by_2rows_drop.append(Y.loc[[i-1]])
Y_2 = by_2rows_drop[0]
for i in range(1,len(by_2rows_drop)):
    Y_2 = pd.concat([Y_2,by_2rows_drop[i]])
Y_2= pd.DataFrame(Y_2)
```

```
feature_by_2rows_drop = []
for i in range(1,len(X)):
    if i%2 == 0:
        feature_by_2rows_drop.append(X.loc[[i-1]])
X_2 = feature_by_2rows_drop[0]
for i in range(1,len(feature_by_2rows_drop)):
    X_2 = pd.concat([X_2,feature_by_2rows_drop[i]])
```

- Zubożanie do 1/3 liczby rekordów w bazie przez pozostawienie co 3 rekordu

```
by_3rows_drop = []
for i in range(1,len(X)):
    if i%3 == 0:
        by_3rows_drop.append(Y.loc[[i-1]])
Y_3 = by_3rows_drop[0]
for i in range(1,len(by_3rows_drop)):
    Y_3 = pd.concat([Y_3,by_3rows_drop[i]])
Y_3= pd.DataFrame(Y_3)
```

```
feature_by_3rows_drop = []
for i in range(1,len(X)):
    if i%3 == 0:
        feature_by_3rows_drop.append(X.loc[[i-1]])
X_3 = feature_by_3rows_drop[0]
for i in range(1,len(feature_by_3rows_drop)):
    X_3 = pd.concat([X_3,feature_by_3rows_drop[i]])
```

	Cycle_Index	dischargeTime	Decrement 3.6-3.4V (s)	Max. Voltage Dischar. (V)	Min. Voltage Charg. (V)	Time at 4.15V (s)	Time constant current (s)	Charging time (s)	RUL	Change
1	1	2595.3	1151.4885	3.67	3.211	5460.001	6755.01	10777.82	1112	1
2	2	7408.64	1172.5125	4.246	3.22	5508.992	6762.02	10500.35	1111	1
3	3	7393.76	1112.99199999999	4.249	3.224	5508.993	6762.02	10420.38	1110	1
4	4	7385.5	1080.32066666667	4.25	3.225	5502.016	6762.02	10322.81	1109	1
5	6	65022.75	29813.487	4.29	3.398	5480.99200000001	53213.54	56699.65	1107	0
...	...	...	...	...	...	...	...	...	...	...
15060	1108	770.44	179.52380952239	3.773	3.742	922.775000000372	1412.38	6678.88	4	1
15061	1109	771.12	179.52380952239	3.773	3.744	915.512000000104	1412.31	6670.38	3	1
15062	1110	769.12	179.357142856345	3.773	3.742	915.512999998406	1412.31	6637.12	2	1
15063	1111	773.88	162.374666666612	3.763	3.839	539.375	1148	7660.62	1	1
15064	1112	677537.27	142740.640000001	4.206	3.305	49680.0040000007	599830.14	599830.14	0	1

Zdjęcie 3: Cała baza danych

	dischargeTime	Decrement 3.6-3.4V (s)	Max. Voltage Dischar. (V)	Min. Voltage Charg. (V)	Time at 4.15V (s)	Time constant current (s)	Charging time (s)
1	2595.30	1151.488500	3.670	3.211	5460.001	6755.01	10777.82
2	7408.64	1172.512500	4.246	3.220	5508.992	6762.02	10500.35
3	7393.76	1112.992000	4.249	3.224	5508.993	6762.02	10420.38
4	7385.50	1080.320667	4.250	3.225	5502.016	6762.02	10322.81
5	65022.75	29813.487000	4.290	3.398	5480.992	53213.54	56699.65
...	...	...	...	...	...	...	...
15060	770.44	179.523810	3.773	3.742	922.775	1412.38	6678.88
15061	771.12	179.523810	3.773	3.744	915.512	1412.31	6670.38
15062	769.12	179.357143	3.773	3.742	915.513	1412.31	6637.12
15063	773.88	162.374667	3.763	3.839	539.375	1148.00	7660.62
15064	677537.27	142740.640000	4.206	3.305	49680.004	599830.14	599830.14

Zdjęcie 4: Kolumny cech

	Change
1	1.0
2	1.0
3	1.0
4	1.0
5	0.0
...	...
15060	1.0
15061	1.0
15062	1.0
15063	1.0
15064	1.0

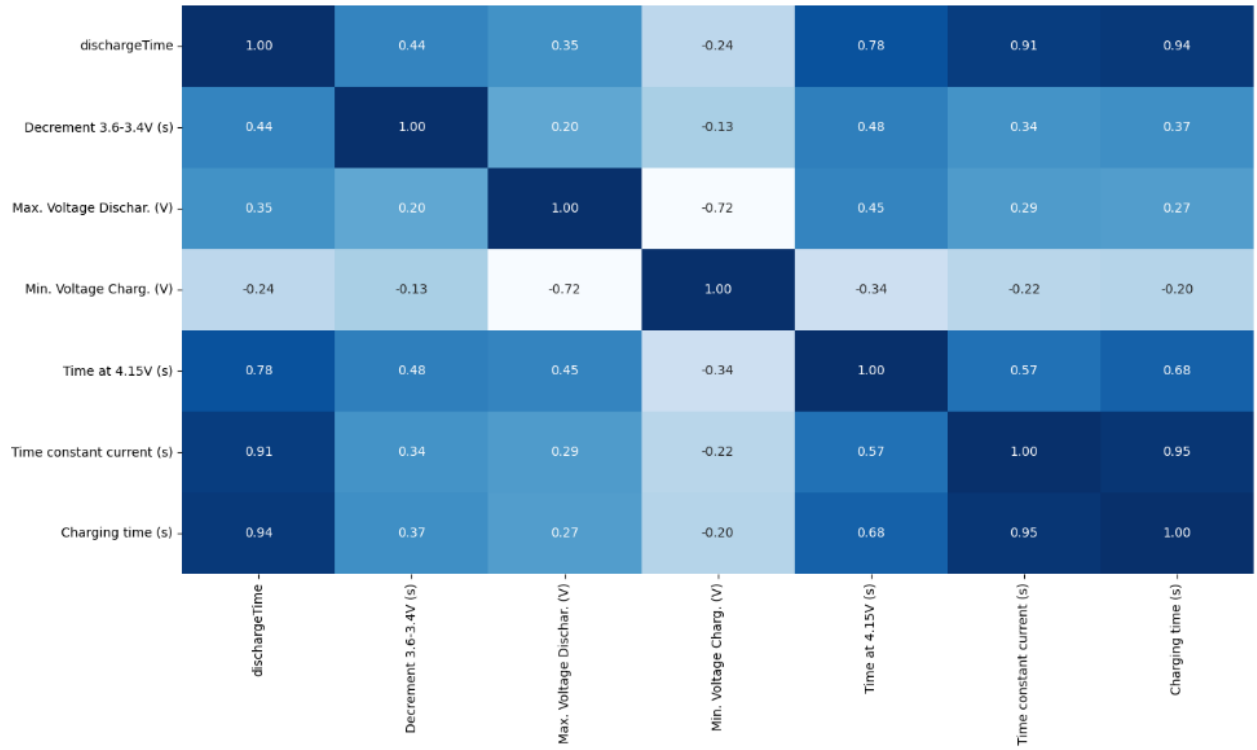
Zdjęcie 5:
Kolumna labeli

2.3. Mapa koleracji pomiędzy cechami

Przedstawienie korelacji pomiędzy poszczególnymi kolumnami cech, wykorzystanie do tego funkcji zawartych w bibliotece seaborn.

```
plt.figure(figsize = (15,8))
sns.heatmap(X.corr(),annot=True, cbar=False, cmap='Blues', fmt='.2f')
```

Zdjęcie 6: Kod przedstawiający mapę korelacji tabeli z kolumnami cech



Zdjęcie 7: Mapa korelacji cech

2.4. Podział na grupę trenigową oraz testową

Grupa testowa to 30% rekordów całej bazy danych.

```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y, test_size=0.30)
last_number_of_nodes = len(X_train)
Y_train = pd.DataFrame(Y_train)
```

Zdjęcie 8: Podział na grupę treningową oraz testową

2.5. Funkcje obliczania Fscore

Funkcje pozwalające na obliczenie i uwzględnienie metryki fscore w trenowanym modelu uczenia maszynowego.

```
def recall_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall

def precision_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision

def f1_m(y_true, y_pred):
    precision = precision_m(y_true, y_pred)
    recall = recall_m(y_true, y_pred)
    return 2*((precision*recall)/(precision+recall+K.epsilon()))
```

Zdjęcie 9: Funkcje pozwalające na obliczenie metryki FScore

2.6. Model sieci uczącej

Prezentowany model posiada 6 warstw aktywacyjnych – 5 typu „relu”, 1 „softmax”. Pierwsza warstwa posiada w zależności od ilości kolumn z cechami, taką ilość wejść, natomiast posiada taką samą ilość wyjść wynikowych. Model optymalizacji został wybrany „Adam” oraz oprócz metryki Fscore, obliczana jest dokładność modelu uczącego.

```
model = Sequential([
    keras.layers.Dense(units=48, input_dim = 7, activation='relu'),
    keras.layers.Dense(units=32, activation='relu'),
    keras.layers.Dense(units=32, activation='relu'),
    keras.layers.Dense(units=128, activation='relu'),
    keras.layers.Dense(units=256, activation='relu'),
    keras.layers.Dense(units=last_number_of_nodes, activation='softmax')
])
model.compile(optimizer='adam',
              loss="sparse_categorical_crossentropy",
              metrics=['accuracy',f1_m,precision_m,recall_m])
```

Zdjęcie 10: Model sieci uczącej z informacji o jej kompilacji

```
model = Sequential([
    keras.layers.Dense(units=14, input_dim = 3, activation='relu'),
    keras.layers.Dense(units=32, activation='relu'),
    keras.layers.Dense(units=32, activation='relu'),
    keras.layers.Dense(units=128, activation='relu'),
    keras.layers.Dense(units=256, activation='relu'),
    keras.layers.Dense(units=last_number_of_nodes, activation='softmax')
])
model.compile(optimizer='adam',
              loss="sparse_categorical_crossentropy",
              metrics=['accuracy',f1_m,precision_m,recall_m])
```

Zdjęcie 11: Model sieci uczącej, zbioru danych zubażałych o co 2 kolumne

```
model = Sequential([
    keras.layers.Dense(units=14, input_dim = 1, activation='relu'),
    keras.layers.Dense(units=32, activation='relu'),
    keras.layers.Dense(units=32, activation='relu'),
    keras.layers.Dense(units=128, activation='relu'),
    keras.layers.Dense(units=256, activation='relu'),
    keras.layers.Dense(units=last_number_of_nodes, activation='softmax')
])
model.compile(optimizer='adam',
              loss="sparse_categorical_crossentropy",
              metrics=['accuracy',f1_m,precision_m,recall_m])
```

Zdjęcie 12: Model sieci uczącej, zbioru danych zubożałych do 1 wartościowej kolumny

2.7. Trenowanie sieci

Trenowanie sieci odbywa się w 2 epokach.

```
history = model.fit(X_train,Y_train, epochs = 2,batch_size=10)
```

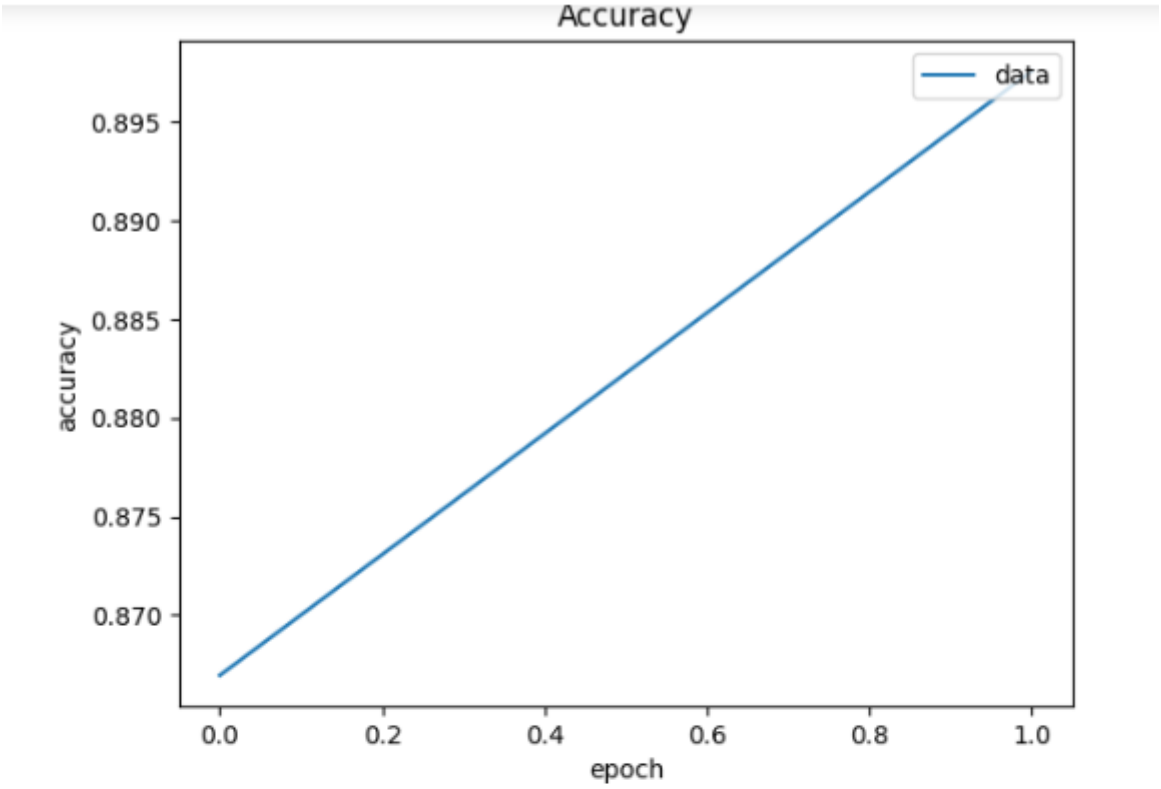
Zdjęcie 13: Trenowanie modelu sieci

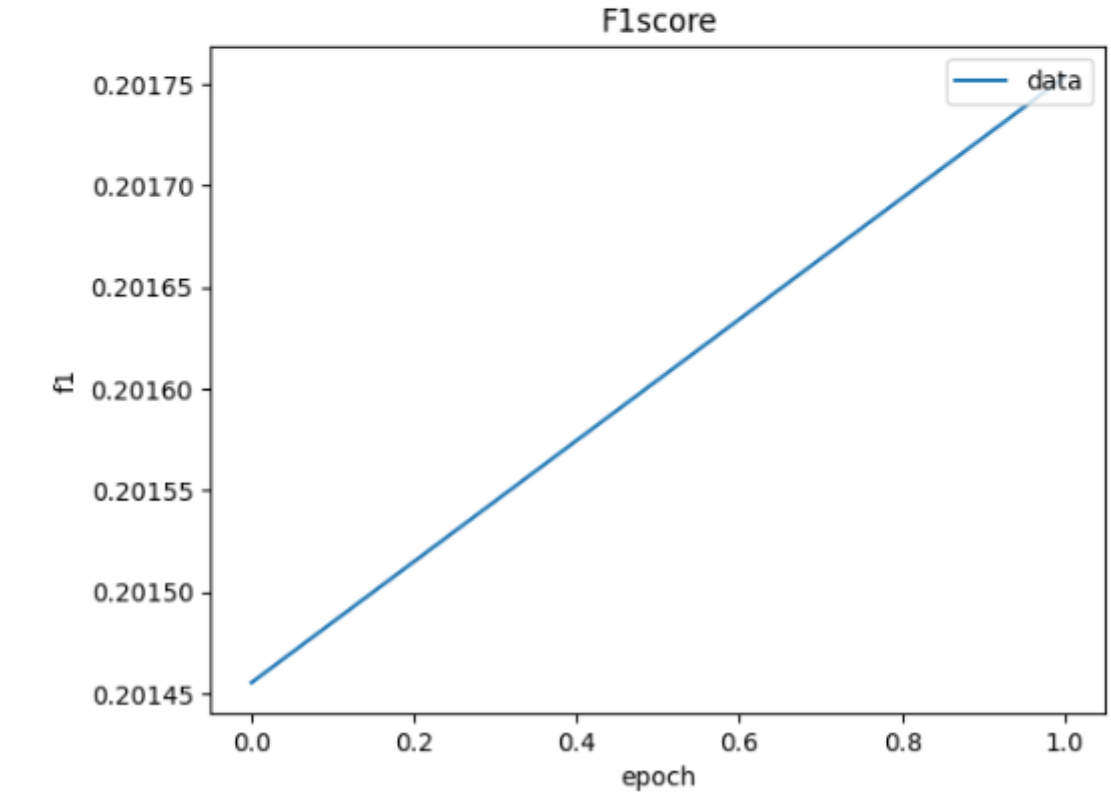
2.8. Prezentacja danych szczegółowych

Prezentacja danych do każdej iteracji zubożania danych (do każdej zmiany ilości kolumn z cechami).

```
plt.plot(history.history['accuracy'])
plt.title('Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['data'], loc='upper right')
plt.show()

plt.plot(history.history['f1_m'])
plt.ylabel('f1')
plt.xlabel('epoch')
plt.legend(['data'], loc='upper right')
plt.title('F1score')
plt.show()

predictions = model.predict(X_test_1)
predictions = predictions[:,0]

predictions = np.where(predictions > 0.5, 1, 0)
false_positive_rate, recall, thresholds = roc_curve(Y_test_1, predictions)
roc_auc = auc(false_positive_rate, recall)
print("AUC: " + str(roc_auc))
```

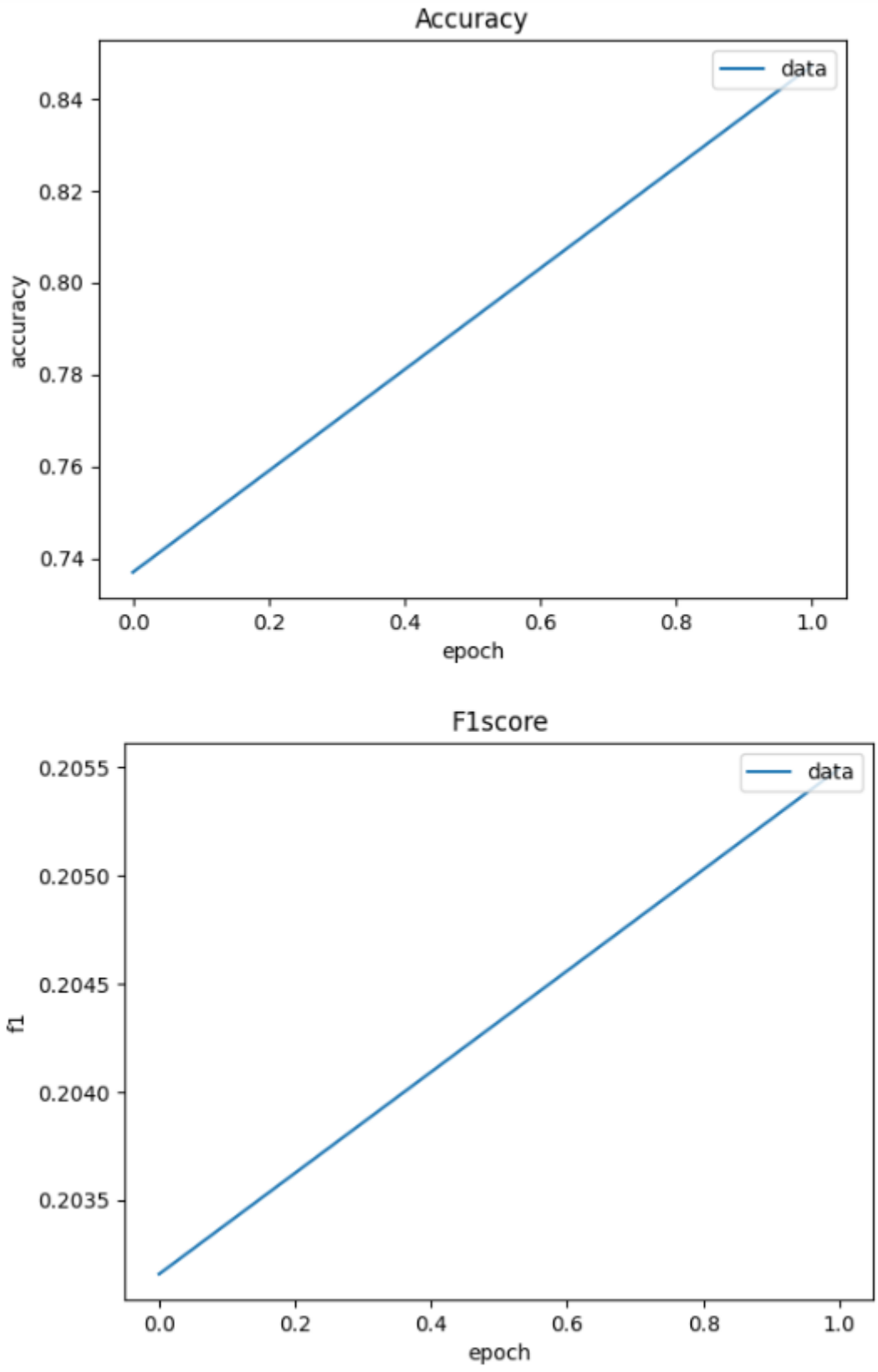
Zdjęcie 14: Prezentacja danych szczegółowych





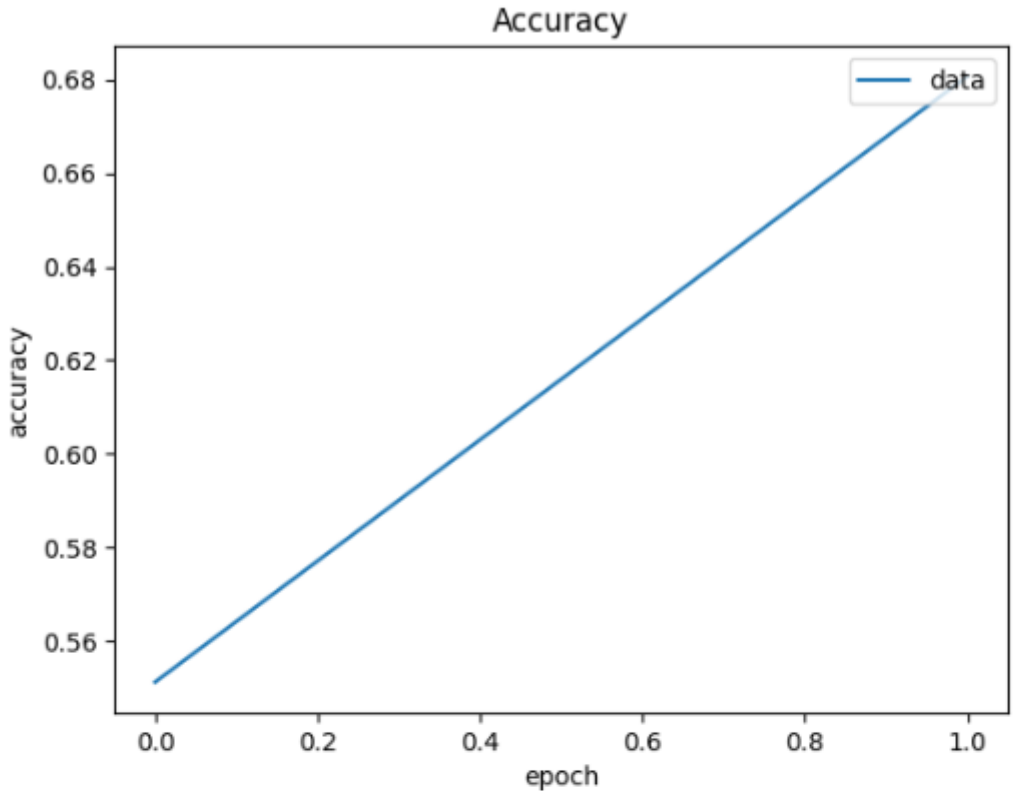

```
142/142 [==============================] - 1s 3ms/step
AUC: 0.38834558867671454
```

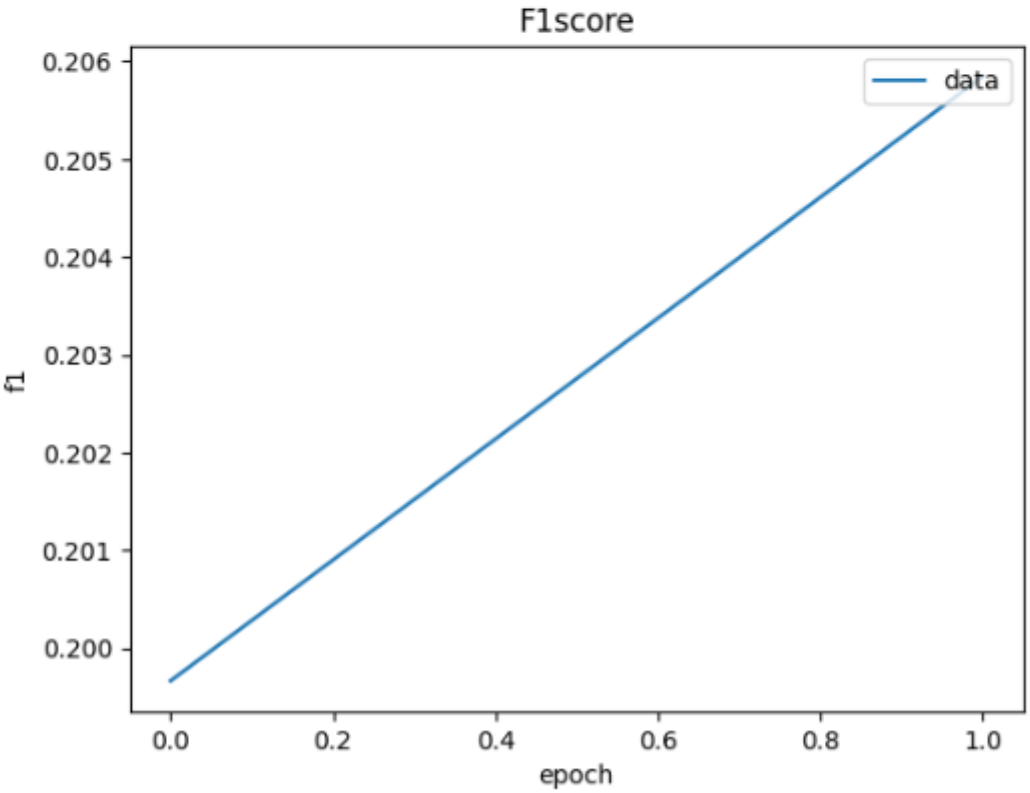
Zdjęcie 15: Wykresy szczegółowe wyniku modelu sieci uczącej dla kolumn z zawartymi wszystkimi cechami




```
142/142 [==============================] - 0s 3ms/step
AUC: 0.48030073924731187
```

Zdjęcie 16: Wykresy szczegółowe wyniku modelu sieci uczącej dla co drugiej kolumny cech






```
48/48 [==============================] - 0s 3ms/step
AUC: 0.4867021276595745
```

Zdjęcie 17: Wykresy szczegółowe wyniku modelu sieci uczącej dla jednej kolumny z cechami

2.9. Dokładność trenowanego modelu

```
X_accuracy_all = np.mean(history.history['accuracy'])
```

```
X_accuracy_by2 = np.mean(history.history['accuracy'])
```

```
X_accuracy_datacut3_disTime=np.mean(history.history['accuracy'])
```

Do powyższych wartości średnich metryki dokładności modelu, utworzony jest wykres przedstawiający ogólny wynik wszystkich iteracji zubażania danych z których możemy wysnuć wnioski odnośnie ile kolumn cech oraz danych potrzebujemy minimalnie, aby stwierdzić, że dana bateria może być jeszcze do użytku.

2.10. Prezentacja wyników i wnioski

```
axis_X = ['all','by2','disTime']
```

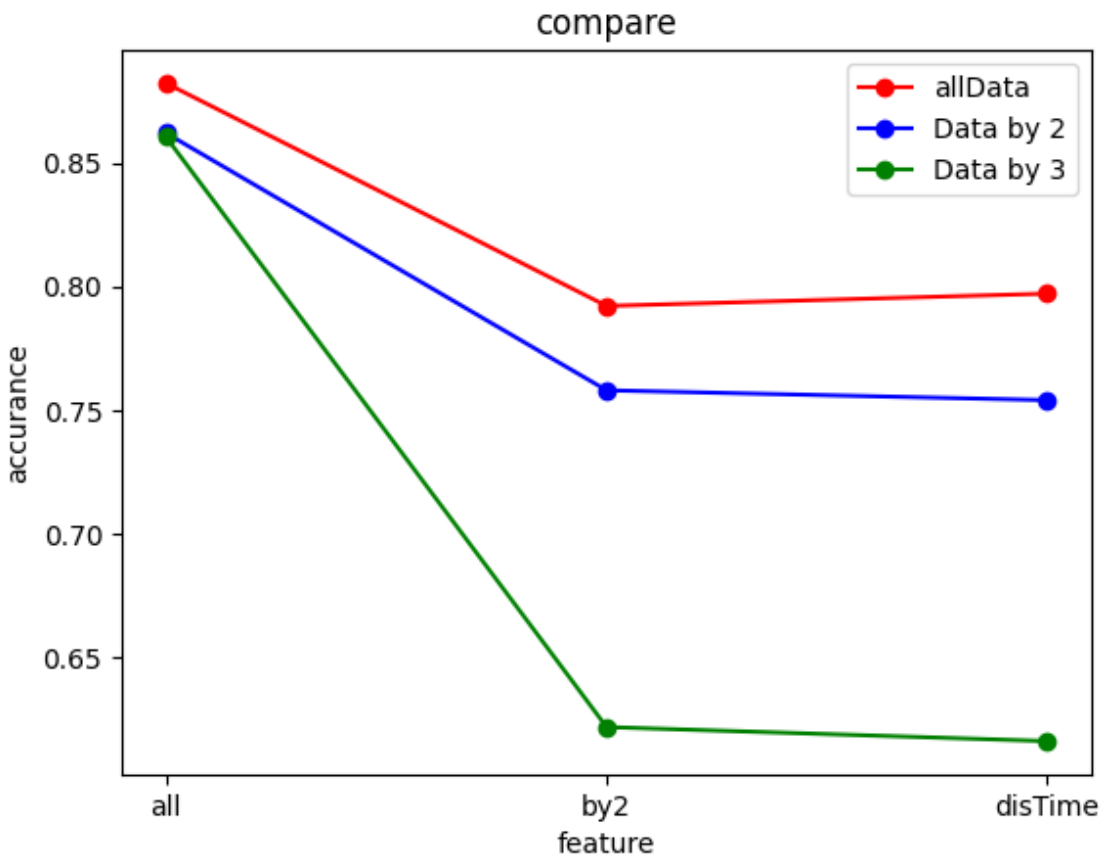
```
allY = [X_accuracy_all, X_accuracy_by2, X_accuracy_disTime]
```

```
allY_2 = [X_accuracy_datacut2_all, X_accuracy_datacut2_by2, X_accuracy_datacut2_disTime]
```

```
allY_3 = [X_accuracy_datacut3_all, X_accuracy_datacut3_by2, X_accuracy_datacut3_disTime]
```

```
plt.plot(axis_X, allY, marker='o', linestyle='-', color='r', label='allData')
plt.plot(axis_X, allY_2, marker='o', linestyle='-', color='b', label='Data by 2')
plt.plot(axis_X, allY_3, marker='o', linestyle='-', color='g', label='Data by 3')
plt.xlabel('feature')
plt.ylabel('accurance')
plt.title('compare')
plt.legend()
plt.show()
```

Powyższy kod pozwala wykonać dany wykres:



W przedstawionym powyżej wykresie widać, że najlepszą dokładność w proporcji do ilości danych widać gdy kolumny z cechami zostały zredukowane co 2 jak i ilość rekordów została zredukowana co druga.

3. Podsumowanie

Przedstawiony powyżej projekt zawierał podstawowe zapoznanie z ekstracją oraz działaniem nad danymi w języku Python, jak i zawierał w sobię część wiedzy teorytycznej jak i praktycznej uczenia maszynowego. Dalszy rozwój i praca nad projektem może skutkować uwzględnienie jej jako pracę inżynierską, a jej wyniki byłyby możliwe do wykorzystania w życiu codziennym np. w magazynach.

Załączniki

Praca umieszczona została na platformie GitHub wraz z plikiem HTML prezentujący cały kod projektu wykonany na jupyter-notebook.

Literatura

[1] https://bolanowski.v.prz.edu.pl/download/word-template-2.html

[2] https://stackoverflow.com/ - rozwiązywanie problemów technicznych

[3] https://towardsdatascience.com/building-our-first-neural-network-in-keras-bdc8abbc17f5