Akademia Ekonomiczno-Humanistyczna w Warszawie

SPRAWOZDANIE

INTELIGENTNA ANALIZA DANYCH

LAB1

DRZEWO DECYZYJNE

23.10.2021

JOANNA PRAJZENDANC

36358

MIŁOSZ SAKOWSKI

36381

Spis treści

1. Cel i przebieg ćwiczenia

Celem ćwiczenia było utrwalenie wiadomości na temat sposobów analizy danych: tworzenie algorytmów klasyfikujących i ich oceny. Podczas ćwiczenia należało pobrać wybraną bazę danych ze strony https://archive.ics.uci.edu/ml/index.php i zaprojektować algorytm klasyfikujący z drzewem decyzyjnym. Potem należało przeanalizować zaproponowany algorytm pod kątem jego dokładności w stosunku do głębokości drzewa decyzyjnego i przedstawić wnioski.

Treść zadania:

Proszę pobrać dowolny zbiór danych ze strony https://archive.ics.uci.edu/ml/index.php

Następnie proszę podzielić zbiór na dane trenujące i testujące, wytrenować 5 modeli drzew decyzyjnych z różną maksymalną głębokością, porównać wyniki. Proszę o sporządzenie sprawozdania z wnioskami.

2. Definicje i założenia

2.1. Wybrana baza danych

Została wybrana baza danych zawierająca informacje na temat pożarów w parku Montensinho: współrzędną X i Y parku gdzie zaczął się ogień, miesiąc i dzień wystąpienia pożaru, współczynniki FFMC, DMC, DC i ISI, które zostały pominięte w ćwiczeniu ze względu na zbytnią złożoność, temperaturę powietrza w °C, współczynnik względnej wilgotności RH, prędkość wiatru w km/h, ilość deszczu w mm/m2 oraz spaloną powierzchnię w ha.

2.2. Założenia algorytmu klasyfikującego

Algorytm przydziela stopień siły pożaru od 0 do 4 na podstawie temperatury powietrza , względnej wilgotności i siły wiatru, według zasad:

- stopień 4 gdy:
  - temperatura większa lub równa sumie mediany i 1/4 różnicy pomiędzy wartością maksymalną a medianą temperatury
  - wilgotność względna mniejsza lub równa 30
  - prędkość wiatru większa lub równa 3.0
- Stopień 3 gdy:
  - temperatura większa lub równa różnicy pomiędzy medianą i 1/4 różnicy pomiędzy wartością maksymalną a medianą temperatury
  - wilgotność względna mniejsza lub równa 50
  - prędkość wiatru większa lub równa 4.0

    lub:
  - temperatura większa lub równa sumie mediany i 1/4 różnicy pomiędzy wartością maksymalną a medianą temperatury
- Stopień 2 gdy:
  - temperatura większa lub równa różnicy pomiędzy miedianą a połową różnicy

pomiędzy wartością maksymalną a medianą temperatury

- wilgotność względna mniejsza lub równa 50
- prędkość wiatru większa lub równa 5.0

- Stopień 1 gdy:
  - temperatura większa lub równa różnicy pomiędzy miedianą a połową różnicy pomiędzy wartością maksymalną a medianą temperatury
- Stopień 0 gdy żadne z powyższych warunków nie zostały spełnione.

2.3. Definicje

W sprawozdaniu będą pojawiać się następujące pojęcia:

- algorytm klasyfikujący - zestaw komend w języku Python, dzięki którym program dokona pożądanych obliczeń; w tym ćwiczeniu dokładniej - skrypt, który przydzieli stopień od 0 do 4 wyrażający
- drzewo decyzyjne - graficzne przedstawienie schematu decyzji (warunków) jakie podejmuje sztuczna inteligencja, aby dopasować predykcje do zadanego modelu
- maksymalna głębokość drzewa decyzyjnego - maksymalna ilość sprawdzanych warunków w jednej iteracji
- dokładność algorytmu - metryka pozwalająca określić jaka część danych została poprawnie zaklasyfikowana
- zbiór danych - zestaw informacji, na których wykonywany jest algorytm klasyfikujący
- zbiór danych trenujących - część początkowego zbioru danych, który posłuży sztucznej inteligencji do „nauki” czyli dopasowanie predykcji
- zbiór danych testujących - część początkowego zbioru danych, która posłuży do przetestowania dokładności predykcji, którą opracowała sztuczna inteligencja na podstawie zbioru trenującego

3. Przedstawienie i analiza zaprojektowanego algorytmu klasyfikującego

3.1. Omówienie kodu krok po kroku

i. Przygotowanie danych

Najpierw zaimportowano dane z pobranego pliku .csv oraz biblioteki pandas oraz tree.

```
import pandas as pd
from sklearn import tree
fires_origin = pd.read_csv("forestfires.csv", header=0, index_col=False)
print(fires_origin.head())
```

```
   X  Y month  day  FFMC   DMC     DC  ISI  temp  RH  wind  rain  area
0  7  5   mar  fri  86.2  26.2   94.3  5.1   8.2  51   6.7   0.0   0.0
1  7  4   oct  tue  90.6  35.4  669.1  6.7  18.0  33   0.9   0.0   0.0
2  7  4   oct  sat  90.6  43.7  686.9  6.7  14.6  33   1.3   0.0   0.0
3  8  6   mar  fri  91.7  33.3   77.5  9.0   8.3  97   4.0   0.2   0.0
4  8  6   mar  sun  89.3  51.3  102.2  9.6  11.4  99   1.8   0.0   0.0
```

Obraz 1: Fragment kodu przedstawiający importowanie danych z pliku

Okazało się, że kolumny z miesiącami i dniami zawierają dane typu string. Ten typ danych nie jest obsługiwany w wybranych metodach użytych w tym ćwiczeniu, dlatego wartości miesięcy i dni zostały zamienione na odpowiedniki typu number.

```
temp_months = fires_origin['month'].copy()
temp_days = fires_origin['day'].copy()
print(temp_months.head(), temp_days.head())
```

```
0    mar
1    oct
2    oct
3    mar
4    mar
Name: month, dtype: object 0    fri
1    tue
2    sat
3    fri
4    sun
Name: day, dtype: object
```

Obraz 2: Fragment kodu pokazujący, że w kolumnach 'month' i 'day' są wartości typu string

```
switcher_months={
            'jan': 1,
            'feb': 2,
            'mar': 3,
            'apr': 4,
            'may': 5,
            'jun': 6,
            'jul': 7,
            'aug': 8,
            'sep': 9,
            'oct': 10,
            'nov': 11,
            'dec': 12,
    }

for i, val in enumerate(temp_months):
    if (type(val) == str):
        temp_months.loc[i] = switcher_months.get(val, 0)
```

```
switcher_days={
            'mon': 1,
            'tue': 2,
            'wed': 3,
            'thu': 4,
            'fri': 5,
            'sat': 6,
            'sun': 7,
        }
for i, val in enumerate(temp_days):
    if (type(val) == str):
        temp_days.loc[i] = switcher_days.get(val, 0)
```

Obraz 3: Fragment kodu przedstawiający sposób podmiany wartości

Następnie utworzono właściwy zbiór danych w formie DataFrame, który posłuży do dalszych etapów ćwiczenia.

```
d = {
    'X': fires_origin['X'].values,
    'Y': fires_origin['Y'].values,
    'month': temp_months.values,
    'day': temp_days.values,
    'temp': fires_origin['temp'].values,
    'RH': fires_origin['RH'].values,
    'wind': fires_origin['wind'].values,
    'rain': fires_origin['rain'].values,
    'area': fires_origin['area'].values
}
fires = pd.DataFrame(d)
print(fires.head())
```

```
   X  Y month day  temp  RH  wind  rain  area
0  7  5     3   5   8.2  51   6.7   0.0   0.0
1  7  4    10   2  18.0  33   0.9   0.0   0.0
2  7  4    10   6  14.6  33   1.3   0.0   0.0
3  8  6     3   5   8.3  97   4.0   0.2   0.0
4  8  6     3   7  11.4  99   1.8   0.0   0.0
```

Obraz 4: Fragment kodu przedstawiający sposób utworzenia ostatecznego zbioru danych

ii. Analiza zbioru danych w celu wyboru warunków algorytmu

Przy projektowaniu algorytmu klasyfikującego wybrano 3 spośród wszystkich parametrów jako główne czynniki wpływające na siłę pożaru: temperatura powietrza, względna wilgotność oraz prędkość wiatru.

Warunki przydzielania stopnia siły pożaru ustalono na podstawie własnych doświadczeń - im cieplejszy dzień, mniejsza wilgotność i mocniejszy wiatr tym większa szansa, że pożar będzie się rozprzestrzeniał.

Wartości graniczne zostały wybrane po zbadaniu mediany, maksimum i minimum każdego z parametrów.

```
import statistics as stat
maximum = max(fires['temp'])
minimum = min(fires['temp'])
median = stat.median(fires['temp'])
diff = round((maximum - median)/4)
print('TEMP MAX:', maximum)
print('TEMP MIN:', minimum)
print('TEMP MEDIAN:', median)
print('TEMP DIFF:', diff)
```

```
TEMP MAX: 33.3
TEMP MIN: 2.2
TEMP MEDIAN: 19.3
TEMP DIFF: 3
```

Obraz 5: Fragment kodu z obliczeniami maksimum, minimum, mediany i różnicy pomiędzy maksimum i medianą dla temperatury

```
max_wind= max(fires['wind'])
min_wind= min(fires['wind'])
median_wind= stat.median(fires['wind'])
print('WIND MAX:', max_wind)
print('WIND MIN:', min_wind)
print('WIND MEDIAN:', median_wind)
```

```
WIND MAX: 9.4
WIND MIN: 0.4
WIND MEDIAN: 4.0
```

Obraz 6: Fragment kodu z obliczeniami maksimum, minimum i mediany prędkości wiatru

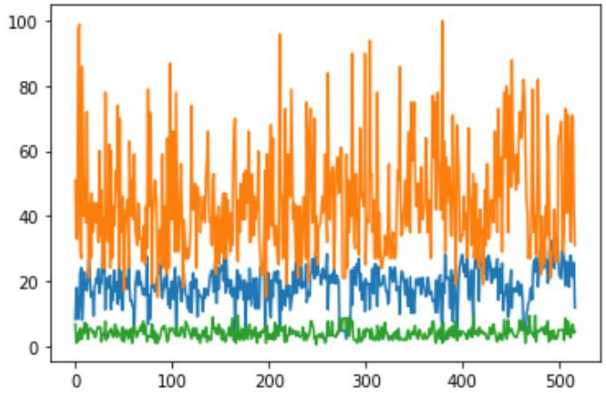
```
max_RH = max(fires['RH'])
min_RH = min(fires['RH'])
median_RH = stat.median(fires['RH'])
print('RH MAX:', max_RH)
print('RH MIN:', min_RH)
print('RH MEDIAN:', median_RH)
```

```
RH MAX: 100
RH MIN: 15
RH MEDIAN: 42
```

Obraz 7: Fragment kodu z obliczeniami maksimum, minimum i mediany względnej wilgotności

```
import matplotlib.pyplot as plt
plt.plot(fires['temp'])
plt.plot(fires['RH'])
plt.plot(fires['wind'])
```

```
[<matplotlib.lines.Line2D at 0x1cc775d82e0>]
```



Obraz 8: Fragment kodu z wykresem temperatury, wilgotności i prędkości wiatru

iii. Algorytm klasyfikujący

Kolejnym etapem było utworzenie zbioru przypisanych stopni siły pożaru na podstawie zaprojektowanego algorytmu klasyfikującego,

```
tp = fires['temp']
rh = fires['RH']
w = fires['wind']
serious = fires['temp'].copy()
for i, val in enumerate(fires['temp']):
    #print(val, rh[i], serious[i])
    if (val >= (median + diff) and rh[i] <= 30 and w[i] >= 3.0):
        serious.loc[i] = 4
    else:
        if (val >= median + diff):
            serious.loc[i] = 3
        else:
            if (val >= (median - diff) and rh[i] <= 50 and w[i] >= 4.0):
                serious.loc[i] = 3
            else:
                if (val >= median - 2*diff and rh[i] <= 50 and w[i] >= 5.0):
                    serious.loc[i] = 2
                else:
                    if (val >= median - 2*diff):
                        serious.loc[i] = 1
                    else:
                        serious.loc[i] = 0
print(serious[1:20])
```

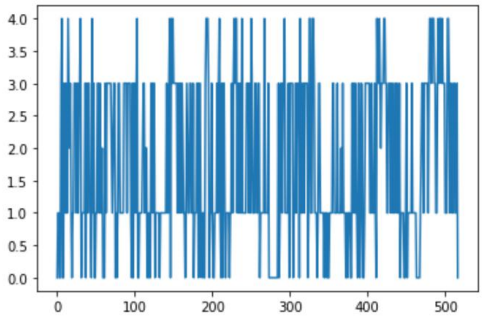
```
1     1.0
2     1.0
3     0.0
4     0.0
5     3.0
6     4.0
7     0.0
8     0.0
9     3.0
10    1.0
11    3.0
12    1.0
13    1.0
14    4.0
15    3.0
16    2.0
17    3.0
18    1.0
19    0.0
Name: temp, dtype: float64
```

Obraz 9: Fragment kodu z algorytmem klasyfikującym

Przy tworzeniu algorytmu klasyfikującego brano również pod uwagę, że zgodnie z rozkładem normalnym najwięcej powinno być wartości na środku, czyli stopni 2 i 3.

```
plt.plot(serious)
```

```
[<matplotlib.lines.Line2D at 0x1cc77626c40>]
```



Obraz 10: Wykres obliczonych wartości siły pożaru

iv. Podział na zbiór trenujący i testujący

W kolejnym kroku podzielono zbiór danych na dwa podzbiory - trenujący i testujący, za pomocą gotowej metody z biblioteki sklearn. Stosunek podziału: 10% zbiór trenujący i 90% zbiór testujący.

```
x = pd.DataFrame(fires.values[:,0:13], columns =['X','Y','month','day','temp', 'RH', 'wind', 'rain', 'area'])
y = pd.DataFrame(serious.values, columns =['SERIOUS'])
```

```
print(x.shape)
print(y.shape)
```

```
(517, 9)
(517, 1)
```

```
print(x[1:5])
print(y[1:5])
```

```
   X  Y month day  temp  RH wind rain area
1  7  4    10   2  18.0  33  0.9  0.0  0.0
2  7  4    10   6  14.6  33  1.3  0.0  0.0
3  8  6     3   5   8.3  97  4.0  0.2  0.0
4  8  6     3   7  11.4  99  1.8  0.0  0.0
   SERIOUS
1      1.0
2      1.0
3      0.0
4      0.0
```

Obraz 11: Fragment kodu przedstawiający przygotowanie zbioru danych do podziału

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.9, random_state=42)
```

```
print(x_train.head())
print(y_train.head())
print(x_test.head())
print(y_test.head())
```

```
     X  Y month day  temp  RH wind rain  area
363  4  3     9   2  15.9  53  2.2  0.0  2.93
50   4  4     9   4  20.8  17  1.3  0.0   0.0
470  5  4     4   7  17.6  27  5.8  0.0   0.0
174  1  4     8   6  14.2  53  1.8  0.0   3.5
445  5  5     8   7  17.3  80  4.5  0.0   0.0
     SERIOUS
363      1.0
50       1.0
470      3.0
174      1.0
445      1.0
     X  Y month day  temp  RH wind rain  area
304  6  5     5   6  11.3  94  4.9  0.0   0.0
501  7  5     8   2  21.6  65  4.9  0.8   0.0
441  8  6     8   1  25.5  29  1.8  0.0  1.23
153  5  4     9   5  20.1  47  4.9  0.0  1.46
503  2  4     8   3  29.2  30  4.9  0.0  1.95
     SERIOUS
304      0.0
501      1.0
441      3.0
153      3.0
503      4.0
```

Obraz 12: Fragment kodu przedstawiający sposób podziału zbioru danych na zbiór trenujący i zbiór testujący

```
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(51, 9)
(51, 1)
(466, 9)
(466, 1)
```

Obraz 13: Fragment kodu przedstawiający liczebność zbiorów po podziale

v. Drzewo decyzyjne z automatyczną głębokością

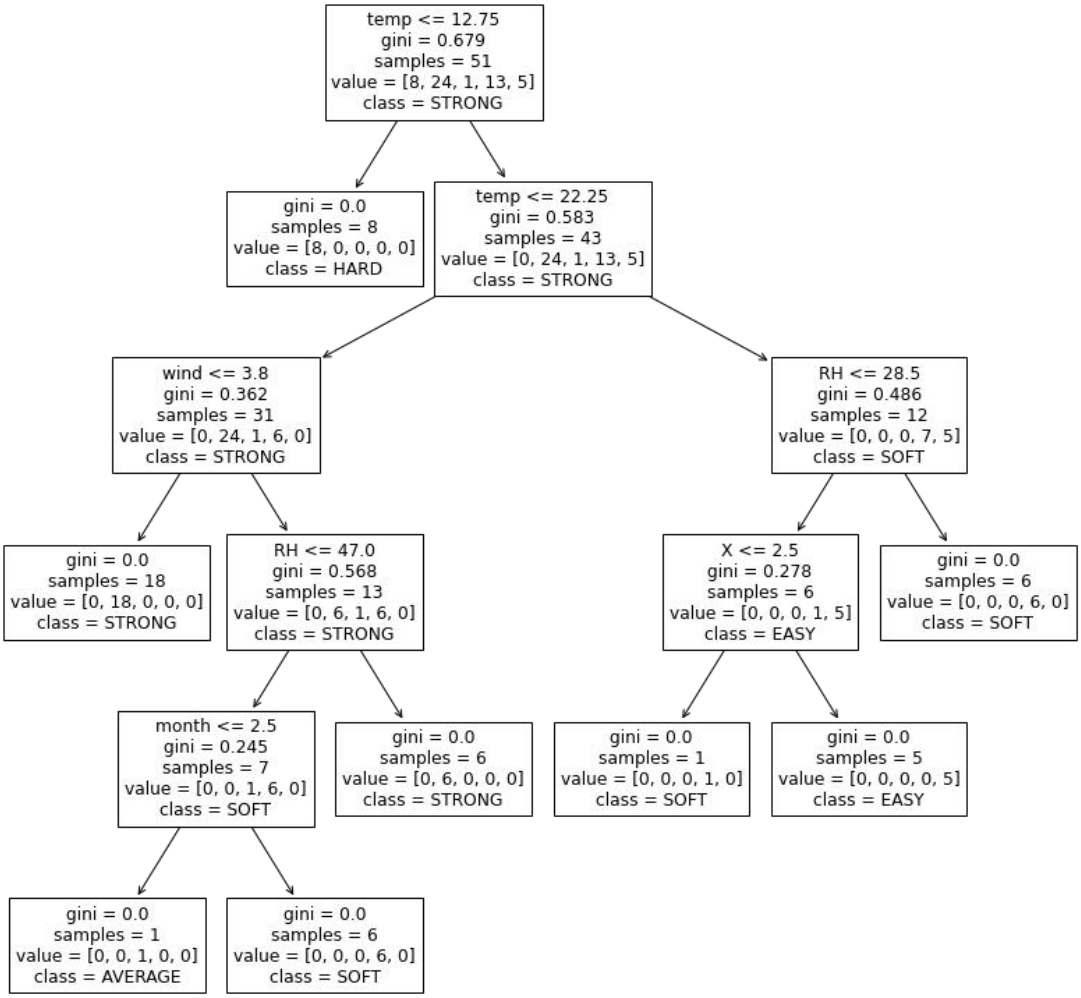
Pierwsze utworzone drzewo decyzyjne zostało utworzone bez określonej maksymalnej głębokości, metoda fit z biblioteki tree miała pełną dowolność.

```
clf = tree.DecisionTreeClassifier()
clf = clf.fit(x_train.values, y_train.values)
```

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(14, 14))
tree.plot_tree(clf, feature_names=list(x.columns), class_names=['HARD', 'STRONG', 'AVERAGE', 'SOFT', 'EASY'])
```

```
[Text(312.48, 697.62, 'temp <= 12.75\ngini = 0.679\nsamples = 51\nvalue = [8, 24, 1, 13, 5]\nclass = STRONG'),
 Text(234.36, 570.78, 'gini = 0.0\nsamples = 8\nvalue = [8, 0, 0, 0, 0]\nclass = HARD'),
 Text(390.6, 570.78, 'temp <= 22.25\ngini = 0.583\nsamples = 43\nvalue = [0, 24, 1, 13, 5]\nclass = STRONG'),
 Text(156.24, 443.94, 'wind <= 3.8\ngini = 0.362\nsamples = 31\nvalue = [0, 24, 1, 6, 0]\nclass = STRONG'),
 Text(78.12, 317.1, 'gini = 0.0\nsamples = 18\nvalue = [0, 18, 0, 0, 0]\nclass = STRONG'),
 Text(234.36, 317.1, 'RH <= 47.0\ngini = 0.568\nsamples = 13\nvalue = [0, 6, 1, 6, 0]\nclass = STRONG'),
 Text(156.24, 190.26, 'month <= 2.5\ngini = 0.245\nsamples = 7\nvalue = [0, 0, 1, 6, 0]\nclass = SOFT'),
 Text(78.12, 63.42000000000007, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1, 0, 0]\nclass = AVERAGE'),
 Text(234.36, 63.42000000000007, 'gini = 0.0\nsamples = 6\nvalue = [0, 0, 0, 6, 0]\nclass = SOFT'),
 Text(312.48, 190.26, 'gini = 0.0\nsamples = 6\nvalue = [0, 6, 0, 0, 0]\nclass = STRONG'),
 Text(624.96, 443.94, 'RH <= 28.5\ngini = 0.486\nsamples = 12\nvalue = [0, 0, 0, 7, 5]\nclass = SOFT'),
 Text(546.84, 317.1, 'X <= 2.5\ngini = 0.278\nsamples = 6\nvalue = [0, 0, 0, 1, 5]\nclass = EASY'),
 Text(468.72, 190.26, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 0, 1, 0]\nclass = SOFT'),
 Text(624.96, 190.26, 'gini = 0.0\nsamples = 5\nvalue = [0, 0, 0, 0, 5]\nclass = EASY'),
 Text(703.08, 317.1, 'gini = 0.0\nsamples = 6\nvalue = [0, 0, 0, 6, 0]\nclass = SOFT')]
```

Obraz 14: Fragment kodu przedstawiający utworzenie drzewa decyzyjnego z automatyczną głębokością



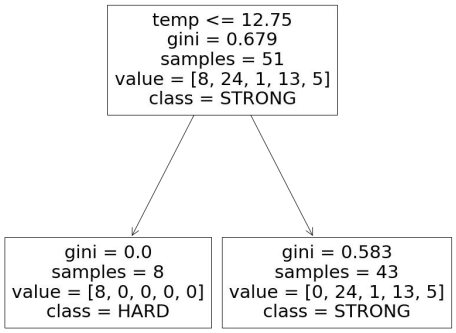
Obraz 15: Podgląd drzewa decyzyjnego z automatyczną głębokością

vi. Drzewo decyzyjne o głębokości maksymalnej = 1

```
clf_max_depth_equal_to_1 = tree.DecisionTreeClassifier(max_depth=1)
clf_max_depth_equal_to_1 = clf_max_depth_equal_to_1.fit(x_train.values, y_train.values)
fig, ax = plt.subplots(figsize=(14, 14))
tree.plot_tree(clf_max_depth_equal_to_1, feature_names=list(x.columns),
               class_names=['HARD', 'STRONG', 'AVERAGE', 'SOFT', 'EASY'])
```

```
[Text(390.6, 570.78, 'temp <= 12.75\ngini = 0.679\nsamples = 51\nvalue = [8, 24, 1, 13, 5]\nclass = STRONG'),
 Text(195.3, 190.26, 'gini = 0.0\nsamples = 8\nvalue = [8, 0, 0, 0, 0]\nclass = HARD'),
 Text(585.9000000000001, 190.26, 'gini = 0.583\nsamples = 43\nvalue = [0, 24, 1, 13, 5]\nclass = STRONG')]
```

Obraz 16: Fragment kodu przedstawiający utworzenie drzewa decyzyjnego o maksymalnej głębokości równej 1



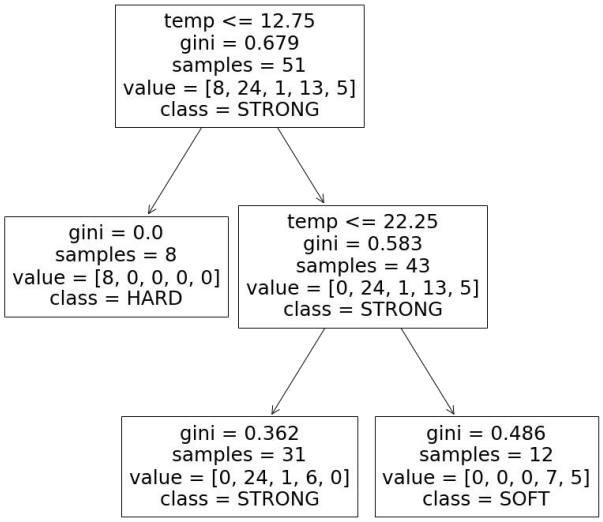
Obraz 17: Podgląd drzewa decyzyjnego o głębokości maksymalnej równej 1

vii. Drzewo decyzyjne o głębokości maksymalnej = 2

```
clf_max_depth_equal_to_2 = tree.DecisionTreeClassifier(max_depth=2)
clf_max_depth_equal_to_2 = clf_max_depth_equal_to_2.fit(x_train.values, y_train.values)
fig, ax = plt.subplots(figsize=(14, 14))
tree.plot_tree(clf_max_depth_equal_to_2, feature_names=list(x.columns),
               class_names=['HARD', 'STRONG', 'AVERAGE', 'SOFT', 'EASY'])
```

```
[Text(312.48, 634.1999999999999, 'temp <= 12.75\ngini = 0.679\nsamples = 51\nvalue = [8, 24, 1, 13, 5]\nclass = STRONG'),
 Text(156.24, 380.52, 'gini = 0.0\nsamples = 8\nvalue = [8, 0, 0, 0, 0]\nclass = HARD'),
 Text(468.72, 380.52, 'temp <= 22.25\ngini = 0.583\nsamples = 43\nvalue = [0, 24, 1, 13, 5]\nclass = STRONG'),
 Text(312.48, 126.84000000000003, 'gini = 0.362\nsamples = 31\nvalue = [0, 24, 1, 6, 0]\nclass = STRONG'),
 Text(624.96, 126.84000000000003, 'gini = 0.486\nsamples = 12\nvalue = [0, 0, 0, 7, 5]\nclass = SOFT')]
```

Obraz 18: Fragment kodu przedstawiający utworzenie drzewa decyzyjnego z maksymalną głębokością równą 2



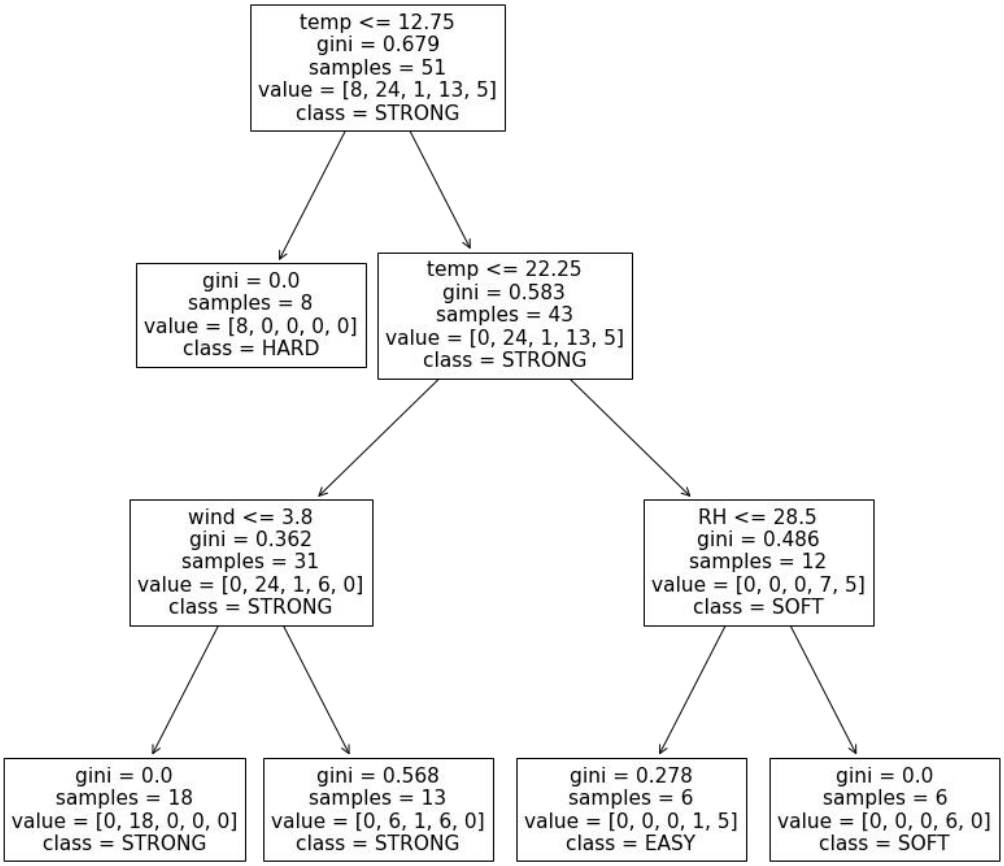
Obraz 19: Podgląd drzewa decyzyjnego o głębokości maksymalnej równej 2

viii. Drzewo decyzyjne o głębokości maksymalnej = 3

```
clf_max_depth_equal_to_3 = tree.DecisionTreeClassifier(max_depth=3)
clf_max_depth_equal_to_3 = clf_max_depth_equal_to_3.fit(x_train.values, y_train.values)
fig, ax = plt.subplots(figsize=(14, 14))
tree.plot_tree(clf_max_depth_equal_to_3, feature_names=list(x.columns),
               class_names=['HARD', 'STRONG', 'AVERAGE', 'SOFT', 'EASY'])
```

```
[Text(292.95000000000005, 665.91, 'temp <= 12.75\ngini = 0.679\nsamples = 51\nvalue = [8, 24, 1, 13, 5]\nclass = STRONG'),
 Text(195.3, 475.65, 'gini = 0.0\nsamples = 8\nvalue = [8, 0, 0, 0, 0]\nclass = HARD'),
 Text(390.6, 475.65, 'temp <= 22.25\ngini = 0.583\nsamples = 43\nvalue = [0, 24, 1, 13, 5]\nclass = STRONG'),
 Text(195.3, 285.39, 'wind <= 3.8\ngini = 0.362\nsamples = 31\nvalue = [0, 24, 1, 6, 0]\nclass = STRONG'),
 Text(97.65, 95.13, 'gini = 0.0\nsamples = 18\nvalue = [0, 18, 0, 0, 0]\nclass = STRONG'),
 Text(292.95000000000005, 95.13, 'gini = 0.568\nsamples = 13\nvalue = [0, 6, 1, 6, 0]\nclass = STRONG'),
 Text(585.9000000000001, 285.39, 'RH <= 28.5\ngini = 0.486\nsamples = 12\nvalue = [0, 0, 0, 7, 5]\nclass = SOFT'),
 Text(488.25, 95.13, 'gini = 0.278\nsamples = 6\nvalue = [0, 0, 0, 1, 5]\nclass = EASY'),
 Text(683.5500000000001, 95.13, 'gini = 0.0\nsamples = 6\nvalue = [0, 0, 0, 6, 0]\nclass = SOFT')]
```

Obraz 20: Fragment kodu przedstawiający utworzenie drzewa decyzyjnego o głębokości maksymalnej równej 3



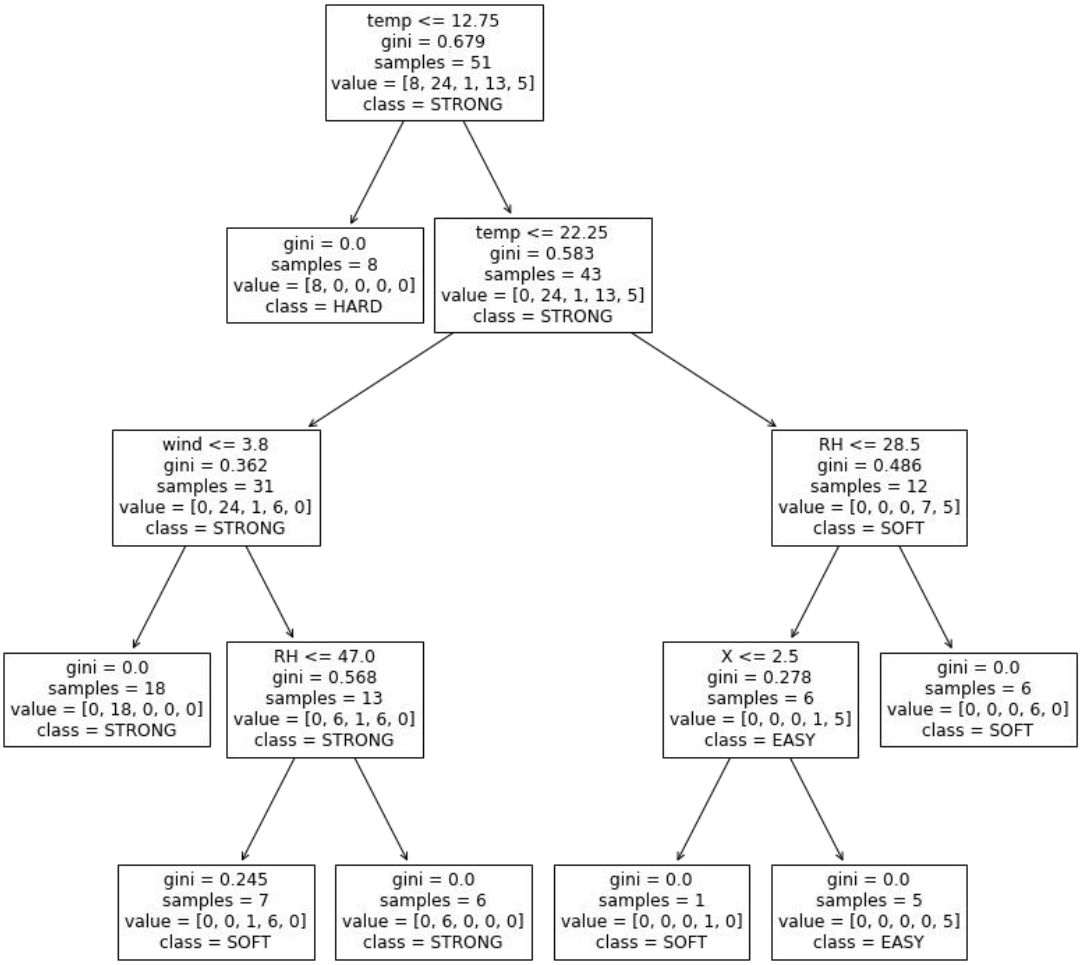
Obraz 21: Podgląd drzewa decyzyjnego o głębokości maksymalnej równej 3

ix. Drzewo decyzyjne o głębokości maksymalnej = 4

```
clf_max_depth_equal_to_4 = tree.DecisionTreeClassifier(max_depth=4)
clf_max_depth_equal_to_4 = clf_max_depth_equal_to_4.fit(x_train.values, y_train.values)
fig, ax = plt.subplots(figsize=(14, 14))
tree.plot_tree(clf_max_depth_equal_to_4, feature_names=list(x.columns),
               class_names=['HARD', 'STRONG', 'AVERAGE', 'SOFT', 'EASY'])
```

```
[Text(312.48, 684.9359999999999, 'temp <= 12.75\ngini = 0.679\nsamples = 51\nvalue = [8, 24, 1, 13, 5]\nclass = STRONG'),
 Text(234.36, 532.728, 'gini = 0.0\nsamples = 8\nvalue = [8, 0, 0, 0, 0]\nclass = HARD'),
 Text(390.6, 532.728, 'temp <= 22.25\ngini = 0.583\nsamples = 43\nvalue = [0, 24, 1, 13, 5]\nclass = STRONG'),
 Text(156.24, 380.52, 'wind <= 3.8\ngini = 0.362\nsamples = 31\nvalue = [0, 24, 1, 6, 0]\nclass = STRONG'),
 Text(78.12, 228.312, 'gini = 0.0\nsamples = 18\nvalue = [0, 18, 0, 0, 0]\nclass = STRONG'),
 Text(234.36, 228.312, 'RH <= 47.0\ngini = 0.568\nsamples = 13\nvalue = [0, 6, 1, 6, 0]\nclass = STRONG'),
 Text(156.24, 76.10399999999993, 'gini = 0.245\nsamples = 7\nvalue = [0, 0, 1, 6, 0]\nclass = SOFT'),
 Text(312.48, 76.10399999999993, 'gini = 0.0\nsamples = 6\nvalue = [0, 6, 0, 0, 0]\nclass = STRONG'),
 Text(624.96, 380.52, 'RH <= 28.5\ngini = 0.486\nsamples = 12\nvalue = [0, 0, 0, 7, 5]\nclass = SOFT'),
 Text(546.84, 228.312, 'X <= 2.5\ngini = 0.278\nsamples = 6\nvalue = [0, 0, 0, 1, 5]\nclass = EASY'),
 Text(468.72, 76.10399999999993, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 0, 1, 0]\nclass = SOFT'),
 Text(624.96, 76.10399999999993, 'gini = 0.0\nsamples = 5\nvalue = [0, 0, 0, 0, 5]\nclass = EASY'),
 Text(703.08, 228.312, 'gini = 0.0\nsamples = 6\nvalue = [0, 0, 0, 6, 0]\nclass = SOFT')]
```

Obraz 22: Fragment kodu przedstawiający utworzenie drzewa decyzyjnego o głębokości maksymalnej równej 4



Obraz 23: Podgląd drzewa decyzyjnego o głębokości maksymalnej równej 4

x. Obliczenie dokładności modelu w zależności od maksymalnej głębokości drzewa decyzyjnego

```
y_test_pred_from_clf = clf.predict(x_test.values)
y_test_pred_from_clf_max_depth_equal_to_1 = clf_max_depth_equal_to_1.predict(x_test.values)
y_test_pred_from_clf_max_depth_equal_to_2 = clf_max_depth_equal_to_2.predict(x_test.values)
y_test_pred_from_clf_max_depth_equal_to_3 = clf_max_depth_equal_to_3.predict(x_test.values)
y_test_pred_from_clf_max_depth_equal_to_4 = clf_max_depth_equal_to_4.predict(x_test.values)
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test, y_test_pred_from_clf))
print(accuracy_score(y_test, y_test_pred_from_clf_max_depth_equal_to_1))
print(accuracy_score(y_test, y_test_pred_from_clf_max_depth_equal_to_2))
print(accuracy_score(y_test, y_test_pred_from_clf_max_depth_equal_to_3))
print(accuracy_score(y_test, y_test_pred_from_clf_max_depth_equal_to_4))
```

```
0.9012875536480687
0.5450643776824035
0.7489270386266095
0.778969957081545
0.9012875536480687
```

Obraz 24: Fragment kodu przedstawiający obliczenia dokładności modelu w zależności od głębokości maksymalnej drzewa decyzyjnego

4. Porównanie wyników

Tabela 1: Wyniki dokładności algorytmu klasyfikującego w zależności od maksymalnej głębokości drzewa decyzyjnego

Maksymalna głębokość drzewa decyzyjnego	Dokładność algorytmu klasyfikującego (zaokrąglona do 2 cyfr po przecinku)
automatyczna	0.90
1	0.55
2	0.75
3	0.78
4	0.90

Porównując dokładność algorytmu można zauważyć, że zbyt mała maksymalna głębokość decyzyjnego może sprawić, że algorytm będzie zbyt niedokładny. Dla maksymalnej głębokości równej 1 dokładność jest na poziomie 50% co jest porównywalne do rzutu monetą. Jednocześnie już dla maksymalnej głębokości równej dwa otrzymano dokładność na poziomie 75%, co w przypadku siły pożaru może być wystarczające np. przy szacowaniu ile potrzeba strażaków do ugaszenia ognia.

Warto również zauważyć, że metoda fit przy braku określonej maksymalnej głębokości drzewa uzyskała bardzo dużą dokładność, na poziomie 90%.

5. Wnioski

Przygotowując zbiór danych, na podstawie którego będą budowane zbiory trenujący i testujący należy bardzo dobrze przemyśleć algorytm klasyfikujący. Dzięki temu można uzyskać zadowalającą dokładność przy mniejszej głębokości maksymalnej drzewa decyzyjnego, co może potencjalnie przyspieszyć późniejsze obliczenia. Jednak gdy bardzo ważna jest duża dokładność algorytmu, warto zwiększyć maksymalną głębokość drzewa decyzyjnego.