

# Raport

April 21, 2022

## 1 Przewidywanie zwycięzcy rundy w grze Counter Strike: Global Offensive

### 1.1 Zmiany 21.04

- dodano rozdział “Wybór atrybutów”
- dodano rozdział “Definicja i inicjalizacja modelu”

### 1.2 Dane

Zbiór danych składa się ze snapshotów rund z około 700 meczów z profesjonalnych turniejów rozgrywanych w 2019 i 2020 roku.

Snapshoty - czyli zestawienie pewnych stanów kluczowych elementów rozgrywki - były rejestrowane podczas gry co 20 sekund aż do rozstrzygnięcia danej rundy. Łączna liczba zapisanych snapshotów wynosi 122411. Część tych rekordów będzie traktowana jako zbiór danych uczących, a pozostała część jako część danych testów. Każdy rekord traktowany jest jako pojedynczy, niezależny element do analizy danych.

### 1.3 Czym jest klasyfikator MLP?

Perceptron wielowarstwowy (MLP) to model sztucznej sieci neuronowej ze sprzężeniem do przodu, który odwzorowuje zestawy danych wejściowych na zestaw odpowiednich danych wyjściowych.

MLP składa się z wielu warstw, a każda warstwa jest w pełni połączona z następną. Węzły warstw to neurony z nieliniowymi funkcjami aktywacji, z wyjątkiem węzłów warstwy wejściowej. Pomiedzy warstwą wejściową a wyjściową może znajdować się jedna lub więcej nieliniowych warstw ukrytych.

### 1.4 Import bibliotek i danych

```
[75]: import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import RobustScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.neural_network import MLPClassifier
from matplotlib import pyplot as plt
import seaborn as sns
```

```
[76]: df = pd.read_csv('Data/csgo_round_snapshots.csv')
```

```
[77]: pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

```
[78]: df.head()
```

```
[78]:
```

	time_left	ct_score	t_score	map	bomb_planted	ct_health	t_health	\
0	175.00	0.0	0.0	de_dust2	False	500.0	500.0	
1	156.03	0.0	0.0	de_dust2	False	500.0	500.0	
2	96.03	0.0	0.0	de_dust2	False	391.0	400.0	
3	76.03	0.0	0.0	de_dust2	False	391.0	400.0	
4	174.97	1.0	0.0	de_dust2	False	500.0	500.0	

	ct_armor	t_armor	ct_money	t_money	ct_helmets	t_helmets	\
0	0.0	0.0	4000.0	4000.0	0.0	0.0	
1	400.0	300.0	600.0	650.0	0.0	0.0	
2	294.0	200.0	750.0	500.0	0.0	0.0	
3	294.0	200.0	750.0	500.0	0.0	0.0	
4	192.0	0.0	18350.0	10750.0	0.0	0.0	

	ct_defuse_kits	ct_players_alive	t_players_alive	ct_weapon_ak47	\
0	0.0	5.0	5.0	0.0	
1	1.0	5.0	5.0	0.0	
2	1.0	4.0	4.0	0.0	
3	1.0	4.0	4.0	0.0	
4	1.0	5.0	5.0	0.0	

	t_weapon_ak47	ct_weapon_aug	t_weapon_aug	ct_weapon_awp	t_weapon_awp	\
0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	

	ct_weapon_bizon	t_weapon_bizon	ct_weapon_cz75auto	t_weapon_cz75auto	\
0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	

	ct_weapon_elite	t_weapon_elite	ct_weapon_famas	t_weapon_famas	\
0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	

4	0.0	0.0	0.0	0.0
---	-----	-----	-----	-----

	ct_weapon_g3sg1	t_weapon_g3sg1	ct_weapon_galilar	t_weapon_galilar	\
0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	

	ct_weapon_glock	t_weapon_glock	ct_weapon_m249	t_weapon_m249	\
0	0.0	5.0	0.0	0.0	
1	0.0	5.0	0.0	0.0	
2	0.0	4.0	0.0	0.0	
3	0.0	3.0	0.0	0.0	
4	0.0	5.0	0.0	0.0	

	ct_weapon_m4a1s	t_weapon_m4a1s	ct_weapon_m4a4	t_weapon_m4a4	\
0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	

	ct_weapon_mac10	t_weapon_mac10	ct_weapon_mag7	t_weapon_mag7	\
0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	

	ct_weapon_mp5sd	t_weapon_mp5sd	ct_weapon_mp7	t_weapon_mp7	\
0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	

	ct_weapon_mp9	t_weapon_mp9	ct_weapon_negev	t_weapon_negev	\
0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	

	ct_weapon_nova	t_weapon_nova	ct_weapon_p90	t_weapon_p90	\
0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	

2	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0

	ct_weapon_r8revolver	t_weapon_r8revolver	ct_weapon_sawedoff	\
0	0.0	0.0	0.0	
1	0.0	0.0	0.0	
2	0.0	0.0	0.0	
3	0.0	0.0	0.0	
4	0.0	0.0	0.0	

	t_weapon_sawedoff	ct_weapon_scar20	t_weapon_scar20	ct_weapon_sg553	\
0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	

	t_weapon_sg553	ct_weapon_ssg08	t_weapon_ssg08	ct_weapon_ump45	\
0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	

	t_weapon_ump45	ct_weapon_xm1014	t_weapon_xm1014	ct_weapon_deagle	\
0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	

	t_weapon_deagle	ct_weapon_fiveseven	t_weapon_fiveseven	ct_weapon_usps	\
0	0.0	0.0	0.0	4.0	
1	0.0	0.0	0.0	4.0	
2	0.0	0.0	0.0	4.0	
3	0.0	0.0	0.0	4.0	
4	0.0	0.0	0.0	4.0	

	t_weapon_usps	ct_weapon_p250	t_weapon_p250	ct_weapon_p2000	\
0	0.0	0.0	0.0	1.0	
1	0.0	0.0	0.0	1.0	
2	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	1.0	

	t_weapon_p2000	ct_weapon_tec9	t_weapon_tec9	ct_grenade_hegrenade	\
--	----------------	----------------	---------------	----------------------	---

0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0
3	1.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0

	t_grenade_hegrenade	ct_grenade_flashbang	t_grenade_flashbang	\
0	0.0	0.0	0.0	
1	0.0	0.0	0.0	
2	0.0	0.0	0.0	
3	0.0	0.0	0.0	
4	0.0	0.0	0.0	

	ct_grenade_smokegrenade	t_grenade_smokegrenade	\
0	0.0	0.0	
1	0.0	2.0	
2	0.0	2.0	
3	0.0	0.0	
4	0.0	0.0	

	ct_grenade_incendiarygrenade	t_grenade_incendiarygrenade	\
0	0.0	0.0	
1	0.0	0.0	
2	0.0	0.0	
3	0.0	0.0	
4	0.0	0.0	

	ct_grenade_molotovgrenade	t_grenade_molotovgrenade	\
0	0.0	0.0	
1	0.0	0.0	
2	0.0	0.0	
3	0.0	0.0	
4	0.0	0.0	

	ct_grenade_decoygrenade	t_grenade_decoygrenade	round_winner
0	0.0	0.0	CT
1	0.0	0.0	CT
2	0.0	0.0	CT
3	0.0	0.0	CT
4	0.0	0.0	CT

```
[79]: df.isnull().sum().sum()
```

```
[79]: 0
```

```
[80]: df.shape
```

```
[80]: (122410, 97)
```

## 1.5 Przetwarzanie danych

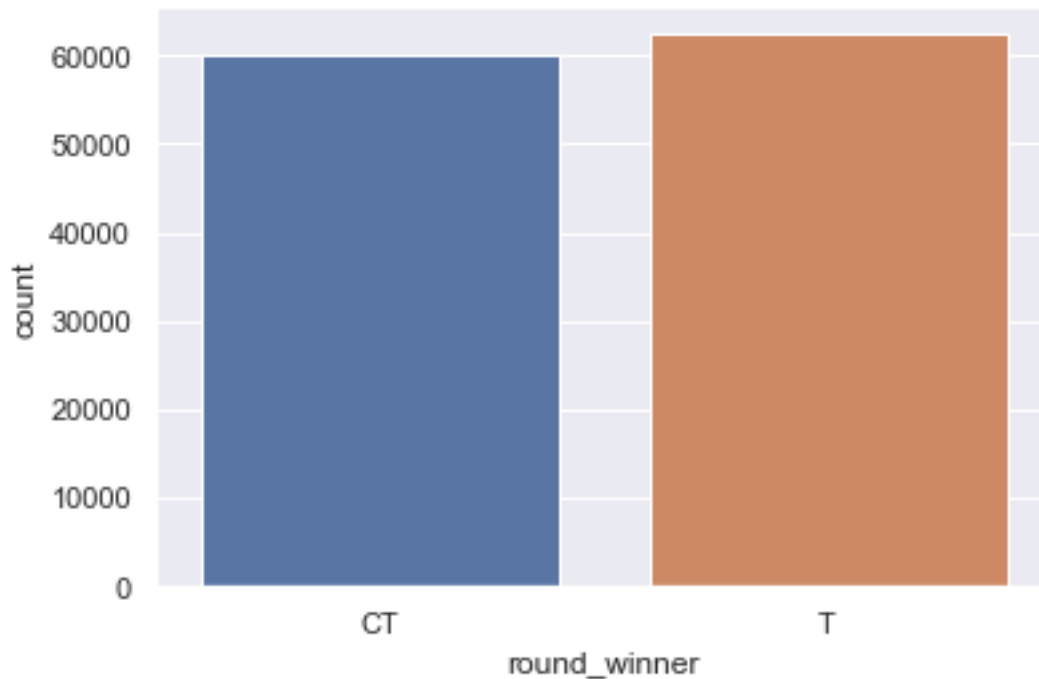
W pierwszej kolejności sprawdzimy licznosc klas.

```
[81]: df['round_winner'].value_counts()
```

```
[81]: T      62406  
     CT      60004  
     Name: round_winner, dtype: int64
```

```
[82]: sns.set_theme(style="darkgrid")  
     sns.countplot(x='round_winner', data=df)
```

```
[82]: <AxesSubplot:xlabel='round_winner', ylabel='count'>
```



Nastepnie przekonwertujemy wszystkie kolumny na wartosci liczbowe.

```
[83]: col = df.drop(df.select_dtypes(np.number), axis = 1).columns  
     col
```

```
[83]: Index(['map', 'bomb_planted', 'round_winner'], dtype='object')
```

```
[84]: lbl = LabelEncoder()
      for value in col:
          df[value] = lbl.fit_transform(df[value])
```

```
[85]: df['bomb_planted'] = df['bomb_planted'].astype(np.int16)
```

```
[86]: cols = [f for f in df.columns if f not in ['round_winner']]
```

Dodatkowo każda wartość zostanie znormalizowana.

```
[87]: scaler = RobustScaler()

      for value in cols:
          scaler = RobustScaler()
          df[value] = scaler.fit_transform(df[[value]])
```

Na koniec podzielimy dane na wektor danych oraz wektor wyników.

```
[88]: x = df.drop(['round_winner'], axis = 1)
      y = df['round_winner']
```

```
[89]: len(cols)
```

```
[89]: 96
```

Finalnie każdy rekord zawiera 96 atrybutów, które prezentują się w sposób przedstawiony poniżej.

```
[90]: x.head()
```

```
[90]:   time_left  ct_score  t_score      map  bomb_planted  ct_health  t_health  \
0   0.715105 -0.857143 -0.857143 -0.666667           0.0   0.000000  0.000000
1   0.545726 -0.857143 -0.857143 -0.666667           0.0   0.000000  0.000000
2   0.010000 -0.857143 -0.857143 -0.666667           0.0  -0.726667 -0.561798
3  -0.168575 -0.857143 -0.857143 -0.666667           0.0  -0.726667 -0.561798
4   0.714837 -0.714286 -0.857143 -0.666667           0.0   0.000000  0.000000

      ct_armor  t_armor  ct_money  t_money  ct_helmets  t_helmets  \
0  -1.291096 -1.136054 -0.112782 -0.191489         -0.5         -0.6
1   0.078767 -0.115646 -0.368421 -0.395137         -0.5         -0.6
2  -0.284247 -0.455782 -0.357143 -0.404255         -0.5         -0.6
3  -0.284247 -0.455782 -0.357143 -0.404255         -0.5         -0.6
4  -0.633562 -1.136054  0.966165  0.218845         -0.5         -0.6

      ct_defuse_kits  ct_players_alive  t_players_alive  ct_weapon_ak47  \
0          -0.333333              0.0              0.0              0.0
1           0.000000              0.0              0.0              0.0
2           0.000000             -1.0             -1.0              0.0
3           0.000000             -1.0             -1.0              0.0
```

4	0.000000	0.0	0.0	0.0
---	----------	-----	-----	-----

	t_weapon_ak47	ct_weapon_aug	t_weapon_aug	ct_weapon_awp	t_weapon_awp \
0	-0.5	0.0	0.0	0.0	0.0
1	-0.5	0.0	0.0	0.0	0.0
2	-0.5	0.0	0.0	0.0	0.0
3	-0.5	0.0	0.0	0.0	0.0
4	-0.5	0.0	0.0	0.0	0.0

	ct_weapon_bizon	t_weapon_bizon	ct_weapon_cz75auto	t_weapon_cz75auto \
0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0

	ct_weapon_elite	t_weapon_elite	ct_weapon_famas	t_weapon_famas \
0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0

	ct_weapon_g3sg1	t_weapon_g3sg1	ct_weapon_galilar	t_weapon_galilar \
0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0

	ct_weapon_glock	t_weapon_glock	ct_weapon_m249	t_weapon_m249 \
0	0.0	0.333333	0.0	0.0
1	0.0	0.333333	0.0	0.0
2	0.0	0.000000	0.0	0.0
3	0.0	-0.333333	0.0	0.0
4	0.0	0.333333	0.0	0.0

	ct_weapon_m4a1s	t_weapon_m4a1s	ct_weapon_m4a4	t_weapon_m4a4 \
0	0.0	0.0	-0.5	0.0
1	0.0	0.0	-0.5	0.0
2	0.0	0.0	-0.5	0.0
3	0.0	0.0	-0.5	0.0
4	0.0	0.0	-0.5	0.0

	ct_weapon_mac10	t_weapon_mac10	ct_weapon_mag7	t_weapon_mag7 \
0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0



2	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0

	ct_weapon_mp5sd	t_weapon_mp5sd	ct_weapon_mp7	t_weapon_mp7	\
0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	

	ct_weapon_mp9	t_weapon_mp9	ct_weapon_negev	t_weapon_negev	\
0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	

	ct_weapon_nova	t_weapon_nova	ct_weapon_p90	t_weapon_p90	\
0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	

	ct_weapon_r8revolver	t_weapon_r8revolver	ct_weapon_sawedoff	\
0	0.0	0.0	0.0	
1	0.0	0.0	0.0	
2	0.0	0.0	0.0	
3	0.0	0.0	0.0	
4	0.0	0.0	0.0	

	t_weapon_sawedoff	ct_weapon_scar20	t_weapon_scar20	ct_weapon_sg553	\
0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	

	t_weapon_sg553	ct_weapon_ssg08	t_weapon_ssg08	ct_weapon_ump45	\
0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	

	t_weapon_ump45	ct_weapon_xm1014	t_weapon_xm1014	ct_weapon_deagle	\
--	----------------	------------------	-----------------	------------------	---

0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0

	t_weapon_deagle	ct_weapon_fiveseven	t_weapon_fiveseven	ct_weapon_usps	\
0	0.0	0.0	0.0	0.333333	
1	0.0	0.0	0.0	0.333333	
2	0.0	0.0	0.0	0.333333	
3	0.0	0.0	0.0	0.333333	
4	0.0	0.0	0.0	0.333333	

	t_weapon_usps	ct_weapon_p250	t_weapon_p250	ct_weapon_p2000	\
0	0.0	0.0	0.0	1.0	
1	0.0	0.0	0.0	1.0	
2	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	1.0	

	t_weapon_p2000	ct_weapon_tec9	t_weapon_tec9	ct_grenade_hegrenade	\
0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	
3	1.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	

	t_grenade_hegrenade	ct_grenade_flashbang	t_grenade_flashbang	\
0	0.0	-0.333333	-0.333333	
1	0.0	-0.333333	-0.333333	
2	0.0	-0.333333	-0.333333	
3	0.0	-0.333333	-0.333333	
4	0.0	-0.333333	-0.333333	

	ct_grenade_smokegrenade	t_grenade_smokegrenade	\
0	-0.333333	-0.333333	
1	-0.333333	0.333333	
2	-0.333333	0.333333	
3	-0.333333	-0.333333	
4	-0.333333	-0.333333	

	ct_grenade_incendiarygrenade	t_grenade_incendiarygrenade	\
0	0.0	0.0	
1	0.0	0.0	
2	0.0	0.0	
3	0.0	0.0	
4	0.0	0.0	

	ct_grenade_molotovgrenade	t_grenade_molotovgrenade \
0	0.0	-0.5
1	0.0	-0.5
2	0.0	-0.5
3	0.0	-0.5
4	0.0	-0.5

	ct_grenade_decoygrenade	t_grenade_decoygrenade
0	0.0	0.0
1	0.0	0.0
2	0.0	0.0
3	0.0	0.0
4	0.0	0.0

### 1.5.1 Wybór atrybutów

W pierwszej kolejności sprawdzimy, które atrybuty mają największy wpływ na klasy.

W tym celu wykorzystamy model *Random Forest Regressor*, który dopasowuje szereg klasyfikujących drzew decyzyjnych do różnych podpróbek zbioru danych i wykorzystuje uśrednianie w celu poprawy dokładności predykcyjnej i kontroli nadmiernego dopasowania.

```
[91]: rf = RandomForestRegressor(n_estimators=150)
      rf.fit(x, y)
```

```
[91]: RandomForestRegressor(n_estimators=150)
```

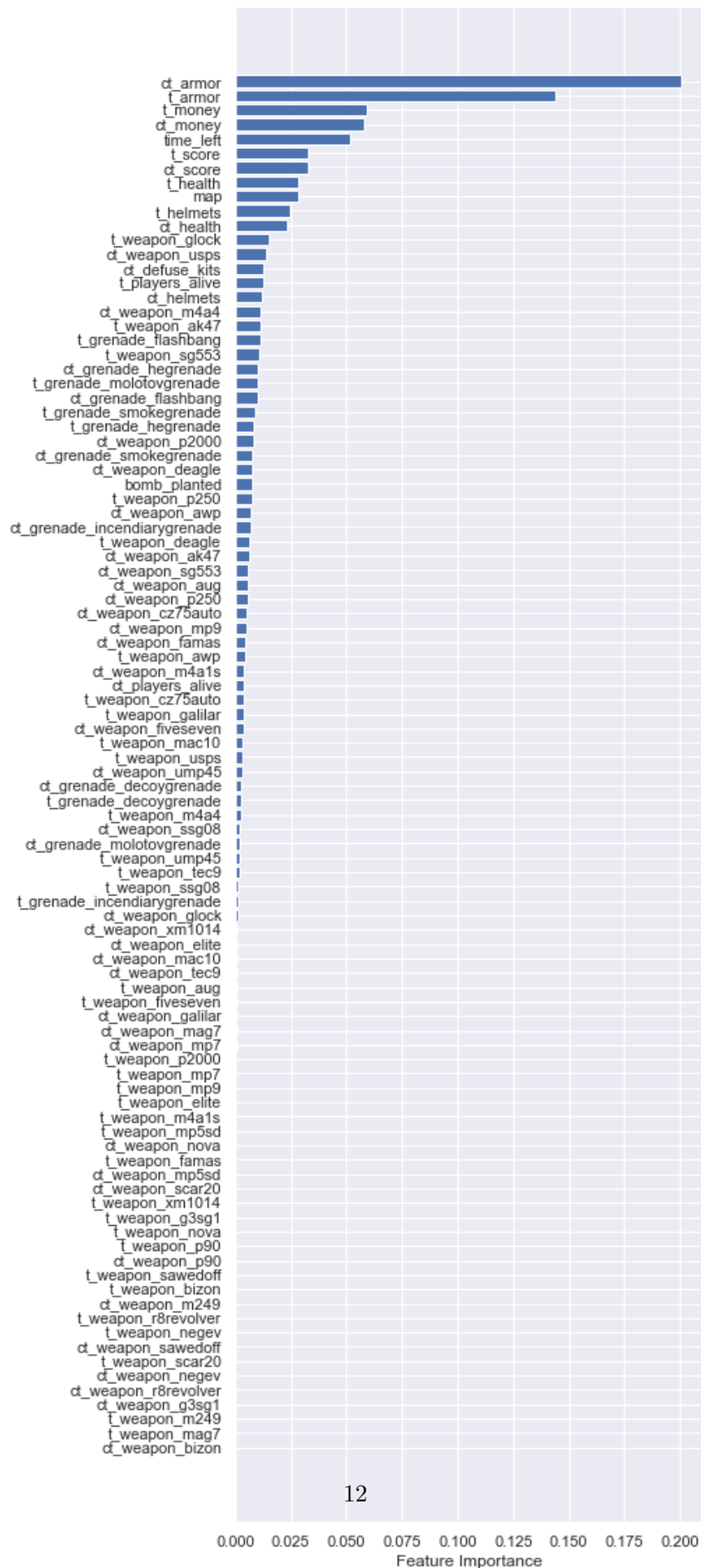
Po stworzeniu i wyuczeniu modelu sortujemy tablicę zawierającą informacje o tym, jak duży wpływ mają konkretne atrybuty w podejmowaniu decyzji.

```
[92]: indexes = rf.feature_importances_.argsort()
```

Wpływ poszczególnych atrybutów został przedstawiony na poniższym wykresie.

```
[93]: plt.figure(figsize=(6,20))
      plt.barh(x.columns[indexes], rf.feature_importances_[indexes])
      plt.xlabel("Feature Importance")
```

```
[93]: Text(0.5, 0, 'Feature Importance')
```



Następnie możemy odrzucić te atrybuty, które mają niski wpływ na podejmowanie decyzji lub nie mają żadnego. W tym celu wybraliśmy te atrybuty, dla których wartość parametru *feature\_importance* jest większa od wartości 0,005.

```
[94]: importances = rf.feature_importances_[indexes]
indexes = indexes[np.argwhere(importances > 0.005)]
indexes = np.concatenate(indexes).ravel().tolist()
columns = x.columns[indexes]
x = x[columns]
```

W tym momencie możemy sprawdzić liczbę atrybutów pozostałych po selekcji.

```
[95]: x.shape
```

```
[95]: (122410, 39)
```

Mamy 38 atrybutów, które prezentują się jak w poniższej tabeli.

```
[96]: x.head()
```

```
[96]:
```

	ct_weapon_mp9	ct_weapon_cz75auto	ct_weapon_p250	ct_weapon_aug	\
0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	

	ct_weapon_sg553	ct_weapon_ak47	t_weapon_deagle	\
0	0.0	0.0	0.0	
1	0.0	0.0	0.0	
2	0.0	0.0	0.0	
3	0.0	0.0	0.0	
4	0.0	0.0	0.0	

	ct_grenade_incendiarygrenade	ct_weapon_awp	t_weapon_p250	bomb_planted	\
0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	

	ct_weapon_deagle	ct_grenade_smokegrenade	ct_weapon_p2000	\
0	0.0	-0.333333	1.0	
1	0.0	-0.333333	1.0	
2	0.0	-0.333333	0.0	
3	0.0	-0.333333	0.0	

4	0.0	-0.333333	1.0	
---	-----	-----------	-----	--

	t_grenade_hegrenade	t_grenade_smokegrenade	ct_grenade_flashbang	\
0	0.0	-0.333333	-0.333333	
1	0.0	0.333333	-0.333333	
2	0.0	0.333333	-0.333333	
3	0.0	-0.333333	-0.333333	
4	0.0	-0.333333	-0.333333	

	t_grenade_molotovgrenade	ct_grenade_hegrenade	t_weapon_sg553	\
0	-0.5	0.0	0.0	
1	-0.5	0.0	0.0	
2	-0.5	0.0	0.0	
3	-0.5	0.0	0.0	
4	-0.5	0.0	0.0	

	t_grenade_flashbang	t_weapon_ak47	ct_weapon_m4a4	ct_helmets	\
0	-0.333333	-0.5	-0.5	-0.5	
1	-0.333333	-0.5	-0.5	-0.5	
2	-0.333333	-0.5	-0.5	-0.5	
3	-0.333333	-0.5	-0.5	-0.5	
4	-0.333333	-0.5	-0.5	-0.5	

	t_players_alive	ct_defuse_kits	ct_weapon_usps	t_weapon_glock	ct_health	\
0	0.0	-0.333333	0.333333	0.333333	0.000000	
1	0.0	0.000000	0.333333	0.333333	0.000000	
2	-1.0	0.000000	0.333333	0.000000	-0.726667	
3	-1.0	0.000000	0.333333	-0.333333	-0.726667	
4	0.0	0.000000	0.333333	0.333333	0.000000	

	t_helmets	map	t_health	ct_score	t_score	time_left	ct_money	\
0	-0.6	-0.666667	0.000000	-0.857143	-0.857143	0.715105	-0.112782	
1	-0.6	-0.666667	0.000000	-0.857143	-0.857143	0.545726	-0.368421	
2	-0.6	-0.666667	-0.561798	-0.857143	-0.857143	0.010000	-0.357143	
3	-0.6	-0.666667	-0.561798	-0.857143	-0.857143	-0.168575	-0.357143	
4	-0.6	-0.666667	0.000000	-0.714286	-0.857143	0.714837	0.966165	

	t_money	t_armor	ct_armor
0	-0.191489	-1.136054	-1.291096
1	-0.395137	-0.115646	0.078767
2	-0.404255	-0.455782	-0.284247
3	-0.404255	-0.455782	-0.284247
4	0.218845	-1.136054	-0.633562

Mając już przetworzone dane, możemy wydzielić z nich zbiór treningowy oraz testowy.

```
[97]: x_train, x_test, y_train, y_test = train_test_split(x, y, stratify = y,
↳test_size = 0.1, random_state = 0)
```

```
[98]: x_train.shape
```

```
[98]: (110169, 39)
```

```
[99]: x_test.shape
```

```
[99]: (12241, 39)
```

Jak widać na podstawie powyższych wywołań, mamy 110169 rekordów w zbiorze treningowym oraz 12241 w zbiorze testowym. W tym momencie możemy rozpocząć inicjalizację modelu.

### 1.5.2 Definicja i inicjalizacja modelu

W celu inicjalizacji modelu zostanie wykorzystany klasyfikator MLP, zaimplementowany w bibliotece *scikit-learn* jako *MLPClassifier*.

*MLPClassifier* określa wielowarstwowy perceptron. W przeciwieństwie do innych algorytmów klasyfikacji, takich jak klasyfikator wektorów wspierających lub naiwny klasyfikator Bayesa, *MLPClassifier* przy wykonaniu zadania klasyfikacji opiera się na podstawowej sieci neuronowej.

Istotne cechy wielowarstwowego perceptronu MLP w bibliotece *scikit-learn*: - w warstwie wyjściowej nie ma funkcji aktywacji, - w przypadku scenariuszy regresji błąd kwadratowy jest funkcją straty, a entropia krzyżowa jest funkcją straty klasyfikacji, - może pracować z regresją pojedynczych, jak i wielu wartości docelowych, - w przeciwieństwie do innych popularnych pakietów, implementacja MLP w *scikit* nie obsługuje GPU. - nie jest możliwe dostrojenie parametrów, takich jak różne funkcje aktywacji, inicjatory wagi itp. dla każdej warstwy.

```
[100]: nn_clf = MLPClassifier(verbose = True)
```

Opcja *verbose=True* powoduje, że na standardowe wyjście będą drukowane wiadomości o postępach w kolejnych iteracjach.

Następnie przechodzimy do dopasowania modelu do macierzy danych *x\_train* i wyniku *y\_train*.

```
[101]: nn_clf.fit(x_train, y_train)
```

```
Iteration 1, loss = 0.48001526
Iteration 2, loss = 0.45443920
Iteration 3, loss = 0.44855651
Iteration 4, loss = 0.44424559
Iteration 5, loss = 0.44119112
Iteration 6, loss = 0.43881870
Iteration 7, loss = 0.43602165
Iteration 8, loss = 0.43413346
Iteration 9, loss = 0.43238168
Iteration 10, loss = 0.43044811
Iteration 11, loss = 0.42913017
```

Iteration 12, loss = 0.42795988  
Iteration 13, loss = 0.42631679  
Iteration 14, loss = 0.42556749  
Iteration 15, loss = 0.42417754  
Iteration 16, loss = 0.42340622  
Iteration 17, loss = 0.42236470  
Iteration 18, loss = 0.42125027  
Iteration 19, loss = 0.42059905  
Iteration 20, loss = 0.41999418  
Iteration 21, loss = 0.41908024  
Iteration 22, loss = 0.41836787  
Iteration 23, loss = 0.41730338  
Iteration 24, loss = 0.41688252  
Iteration 25, loss = 0.41585448  
Iteration 26, loss = 0.41513754  
Iteration 27, loss = 0.41442731  
Iteration 28, loss = 0.41370405  
Iteration 29, loss = 0.41292838  
Iteration 30, loss = 0.41222901  
Iteration 31, loss = 0.41179587  
Iteration 32, loss = 0.41053177  
Iteration 33, loss = 0.41051793  
Iteration 34, loss = 0.40950201  
Iteration 35, loss = 0.40865221  
Iteration 36, loss = 0.40824867  
Iteration 37, loss = 0.40808898  
Iteration 38, loss = 0.40742771  
Iteration 39, loss = 0.40700306  
Iteration 40, loss = 0.40642847  
Iteration 41, loss = 0.40590989  
Iteration 42, loss = 0.40552802  
Iteration 43, loss = 0.40446347  
Iteration 44, loss = 0.40421381  
Iteration 45, loss = 0.40373333  
Iteration 46, loss = 0.40351673  
Iteration 47, loss = 0.40262809  
Iteration 48, loss = 0.40218768  
Iteration 49, loss = 0.40213368  
Iteration 50, loss = 0.40121764  
Iteration 51, loss = 0.40075708  
Iteration 52, loss = 0.40043096  
Iteration 53, loss = 0.40009568  
Iteration 54, loss = 0.39973165  
Iteration 55, loss = 0.39895798  
Iteration 56, loss = 0.39864603  
Iteration 57, loss = 0.39861160  
Iteration 58, loss = 0.39762831  
Iteration 59, loss = 0.39733803



Iteration 60, loss = 0.39716891  
Iteration 61, loss = 0.39704845  
Iteration 62, loss = 0.39649278  
Iteration 63, loss = 0.39578075  
Iteration 64, loss = 0.39549061  
Iteration 65, loss = 0.39521812  
Iteration 66, loss = 0.39522164  
Iteration 67, loss = 0.39524714  
Iteration 68, loss = 0.39422611  
Iteration 69, loss = 0.39430835  
Iteration 70, loss = 0.39356381  
Iteration 71, loss = 0.39346180  
Iteration 72, loss = 0.39315167  
Iteration 73, loss = 0.39303061  
Iteration 74, loss = 0.39191398  
Iteration 75, loss = 0.39239674  
Iteration 76, loss = 0.39221024  
Iteration 77, loss = 0.39189102  
Iteration 78, loss = 0.39154758  
Iteration 79, loss = 0.39088792  
Iteration 80, loss = 0.39112278  
Iteration 81, loss = 0.39100867  
Iteration 82, loss = 0.39028134  
Iteration 83, loss = 0.39076824  
Iteration 84, loss = 0.39061000  
Iteration 85, loss = 0.38938957  
Iteration 86, loss = 0.38998950  
Iteration 87, loss = 0.38922879  
Iteration 88, loss = 0.38925665  
Iteration 89, loss = 0.38904180  
Iteration 90, loss = 0.38847561  
Iteration 91, loss = 0.38872945  
Iteration 92, loss = 0.38802789  
Iteration 93, loss = 0.38779857  
Iteration 94, loss = 0.38791565  
Iteration 95, loss = 0.38757272  
Iteration 96, loss = 0.38691663  
Iteration 97, loss = 0.38706029  
Iteration 98, loss = 0.38731323  
Iteration 99, loss = 0.38713128  
Iteration 100, loss = 0.38632400  
Iteration 101, loss = 0.38625907  
Iteration 102, loss = 0.38662615  
Iteration 103, loss = 0.38597629  
Iteration 104, loss = 0.38603507  
Iteration 105, loss = 0.38558089  
Iteration 106, loss = 0.38536262  
Iteration 107, loss = 0.38530541

Iteration 108, loss = 0.38565438  
Iteration 109, loss = 0.38476153  
Iteration 110, loss = 0.38488125  
Iteration 111, loss = 0.38453152  
Iteration 112, loss = 0.38426963  
Iteration 113, loss = 0.38432219  
Iteration 114, loss = 0.38358462  
Iteration 115, loss = 0.38421846  
Iteration 116, loss = 0.38439748  
Iteration 117, loss = 0.38364193  
Iteration 118, loss = 0.38367817  
Iteration 119, loss = 0.38379879  
Iteration 120, loss = 0.38321187  
Iteration 121, loss = 0.38320733  
Iteration 122, loss = 0.38313328  
Iteration 123, loss = 0.38307715  
Iteration 124, loss = 0.38343239  
Iteration 125, loss = 0.38256299  
Iteration 126, loss = 0.38295389  
Iteration 127, loss = 0.38204827  
Iteration 128, loss = 0.38201784  
Iteration 129, loss = 0.38197456  
Iteration 130, loss = 0.38166846  
Iteration 131, loss = 0.38219273  
Iteration 132, loss = 0.38180104  
Iteration 133, loss = 0.38191366  
Iteration 134, loss = 0.38152079  
Iteration 135, loss = 0.38189897  
Iteration 136, loss = 0.38171513  
Iteration 137, loss = 0.38111496  
Iteration 138, loss = 0.38082634  
Iteration 139, loss = 0.38107588  
Iteration 140, loss = 0.38081955  
Iteration 141, loss = 0.38068893  
Iteration 142, loss = 0.38123919  
Iteration 143, loss = 0.37999542  
Iteration 144, loss = 0.38015388  
Iteration 145, loss = 0.38021811  
Iteration 146, loss = 0.38046293  
Iteration 147, loss = 0.37932430  
Iteration 148, loss = 0.37970658  
Iteration 149, loss = 0.37992756  
Iteration 150, loss = 0.37986059  
Iteration 151, loss = 0.37991824  
Iteration 152, loss = 0.37906741  
Iteration 153, loss = 0.37931127  
Iteration 154, loss = 0.37918259  
Iteration 155, loss = 0.37899981

Iteration 156, loss = 0.37897268  
Iteration 157, loss = 0.37822194  
Iteration 158, loss = 0.37952528  
Iteration 159, loss = 0.37870348  
Iteration 160, loss = 0.37887201  
Iteration 161, loss = 0.37843006  
Iteration 162, loss = 0.37825528  
Iteration 163, loss = 0.37847671  
Iteration 164, loss = 0.37797510  
Iteration 165, loss = 0.37796924  
Iteration 166, loss = 0.37800517  
Iteration 167, loss = 0.37818095  
Iteration 168, loss = 0.37811403  
Iteration 169, loss = 0.37720168  
Iteration 170, loss = 0.37786452  
Iteration 171, loss = 0.37786796  
Iteration 172, loss = 0.37730313  
Iteration 173, loss = 0.37780171  
Iteration 174, loss = 0.37751012  
Iteration 175, loss = 0.37790143  
Iteration 176, loss = 0.37707576  
Iteration 177, loss = 0.37698748  
Iteration 178, loss = 0.37658086  
Iteration 179, loss = 0.37602368  
Iteration 180, loss = 0.37696320  
Iteration 181, loss = 0.37700843  
Iteration 182, loss = 0.37714355  
Iteration 183, loss = 0.37715894  
Iteration 184, loss = 0.37655213  
Iteration 185, loss = 0.37635376  
Iteration 186, loss = 0.37616180  
Iteration 187, loss = 0.37584692  
Iteration 188, loss = 0.37625805  
Iteration 189, loss = 0.37534958  
Iteration 190, loss = 0.37631373  
Iteration 191, loss = 0.37611275  
Iteration 192, loss = 0.37556852  
Iteration 193, loss = 0.37672967  
Iteration 194, loss = 0.37621180  
Iteration 195, loss = 0.37603356  
Iteration 196, loss = 0.37572429  
Iteration 197, loss = 0.37596753  
Iteration 198, loss = 0.37518520  
Iteration 199, loss = 0.37554752  
Iteration 200, loss = 0.37548067

C:\Users\adasi\AppData\Local\pypoetry\Cache\virtualenvs\csgo-round-prediction-  
GhoYBn2B-py3.10\lib\site-

```
packages\sklearn\normal_network\_multilayer_perceptron.py:692:  
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and  
the optimization hasn't converged yet.  
    warnings.warn(  

```

```
[101]: MLPClassifier(verbose=True)
```

W tym momencie możemy sprawdzić średnią trafność modelu na rozważanych danych testowych.

```
[102]: print(f'Model sieci neuronowej: {nn_clf.score(x_test, y_test)}')
```

Model sieci neuronowej: 0.7863736622824933

## 1.6 Podział prac

1. Wybór niezbędnych bibliotek – Jakub Michalak
2. Walidacja i wydzielenie danych treningowych – Damian Opoka
3. Analiza i przygotowanie danych – Damian Opoka
4. Definicja modelu – Adam Ryl
5. Weryfikacja krzyżowa - Adam Ryl
6. Dostosowanie hiperparametrów - Jakub Michalak
7. Wygenerowanie wyników - Piotr Kryczka
8. Analiza wyników - Piotr Kryczka