Politechnika Świętokrzyska w Kielcach

Wydział Elektroniki, Automatyki i Informatyki

Projekt: **Analiza i wizualizacja danych**

Grupa: **1ID21A**	Temat: **Analiza statystyczna i eksploracyjna danych rankingowych z meczów rozegranych w grze Counter-Strike: Global Offensive**	Skład grupy: **Michał Młodawski**
Rok studiów: **5**		

Temat projektu i zakres prac

Tematem projektu było przygotowanie aplikacji służącej do przeprowadzenie analizy statystycznej i eksploracyjnej zestawu danych pozyskanych z zapisów rozgrywek meczów rankingowych z gry Counter-Strike: Global Offensive.

Całość wykonał Michał Młodawski.

Cel projektu i jego opis

Celem projektu było przygotowanie aplikacji służącej do przeprowadzenie analizy statystycznej i eksploracyjnej zestawu danych pozyskanych z zapisów rozgrywek meczów rankingowych z gry Counter-Strike: Global Offensive.

Część projektu odpowiedzialna za analizę statystyczną posiada poniżą funkcjonalność:

- Importowanie danych z pliku CSV do bazy danych
- Wyznaczanie zmiennych statystycznych takich jak:
    - Wartość minimalna, maksymalna, średnia
    - Kwantyl rzędu Q1, Q2, Q3
    - Kwantyl 0.1 i 0.9
    - Rozstęp międzykwartylowy IQR
    - Punkt oddalony poniżej i punkt oddalony powyżej
- Wyznaczenie współczynnika korelacji liniowej Pearsona dla 6 argumentów
- Wyznaczenie prostej regresji liniowej pomiędzy wybranymi parami zmiennych
- Wizualizacja danych w postaci histogramu oraz wykresu kołowego

Część projektu odpowiedzialna za analizę eksploracyjną posiada funkcjonalność testowania sieci neuronowych dla jednej z wybranych funkcji inicjalizacji wag, określonego typu aktywacji neuronu, a także ilości epok. Jako informacje zwrotną zostaje wygenerowany komunikat w którym możemy dowiedzieć się o dokładności etapu uczenia, precyzji danych, poziom ufności modelu, a także jego ocenę ewaluacji podczas testów. Komunikat zwraca także tablicę pomyłek w formie tabeli i tablicę pomyłek w formie zapisu macierzowego.

Przegląd literatury

1. [https://www.youtube.com/watch?v=aircAruvnKk] Dostęp z dnia: 01.01.2021 - cała seria
2. [https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall] Dostęp z dnia: 10.01.2021
3. [https://developers.google.com/machine-learning/crash-course/classification/accuracy] Dostęp z dnia: 10.01.2021
4. [https://developers.google.com/machine-learning/crash-course/classification/true-false-positive-negative] Dostęp z dnia: 10.01.2021
5. [https://www.youtube.com/watch?v=2-VEwB8n7Fo] Dostęp z dnia: 10.01.2021
6. [https://youtu.be/AZborBAKNjs] Dostęp z dnia: 10.01.2021

Opis danych wybranych do analizy

Zbiór danych składa się z zapisu rund z około 700 meczów z turniejów rozegranym poziomie mistrzostw świata w 2019 i 2020 roku. Rundy rozgrzewkowe i restarty zostały odfiltrowane. Następnie zbiór danych zostatał wstępnie przetworzony i spłaszczony, aby poprawić czytelność i ułatwić algorytmom przetwarzanie danych. Całkowita liczba migawek wynosi 122411. Zbiór danych posiada aż 97 atrybutów w tym takie dane jak która drużyna wygrała rundę, czas pozostały do zakończenia rundy, wynik drużyny terrorystów (ilość zdobytych punktów w poprzednich rundach) i wynik drużyny antyterrorystów. Ilość życia i poziom pancerza dla poszczególnych drużyn. Mapa na której rozegrano mecz, ilość gotówki dla obu drużyn, a także czy członkowie drużyn posiadali hełm na swoim uzbrojeniu oraz czy bomba została uzbrojona. Oprócz tych danych zbiór posiada atrybuty w których zliczono liczbę uzbrojenia jaką posiadały drużyny w czasie rundy.

Opis zastosowanych technologii

Projekt został podzielony na dwie osobne aplikacje, obie zostały napisany w języku Java i wykorzystywały architekturę REST do komunikacji z warstwą prezentacji. Aplikacje zostały wyposażone w interfejs graficzny, który został napisany w języku HTML z wykorzystaniem JavaScript i frameworku vue.js z biblioteką axios do komunikacji z warstwą serwerową. Za architekturę REST odpowiada Srping Web, który umożliwia odpowiednią konfigurację endpointów. Do wykresów została wykorzystana biblioteka chart.js. Związku z wykorzystaniem bazy danych MySQL został wykorzystany pakiet Srping Data z Spring Data Repository w celu stworzenia prostej komunikacji między bazą danych, a aplikacją dzięki wykorzystaniu repozytoriów JPA. Proces obsługi metod statystycznych był wspomagany

poprzez wykonywanie zapytań SQL w tym takich funkcji jak min, max, avg, sum, power. A także wykorzystywanie funkcji matematycznych dostępnych z poziomu języka Java przykładowo Math.pow. Proces analizy danych eksploracyjnych został zaimplementowany z wykorzystaniem biblioteki deeplearning4j. Ta biblioteka umożliwia wstępną obróbkę danych poprzez mechanizm normalizacji, a także przygotowanie od bardzo prosty do niezwykle skomplikowanych sieci neuronowych modelu klasyfikacji danych i regresji danych. W moim projekcie wykorzystałem metodę MultiLayerConfiguration dostępną w powyższej bibliotece. Dane zostały wcześniej przygotowane do formatu CSV w celu dalszej normalizacji.

Projekt procesu analizy statystycznej i implementacja

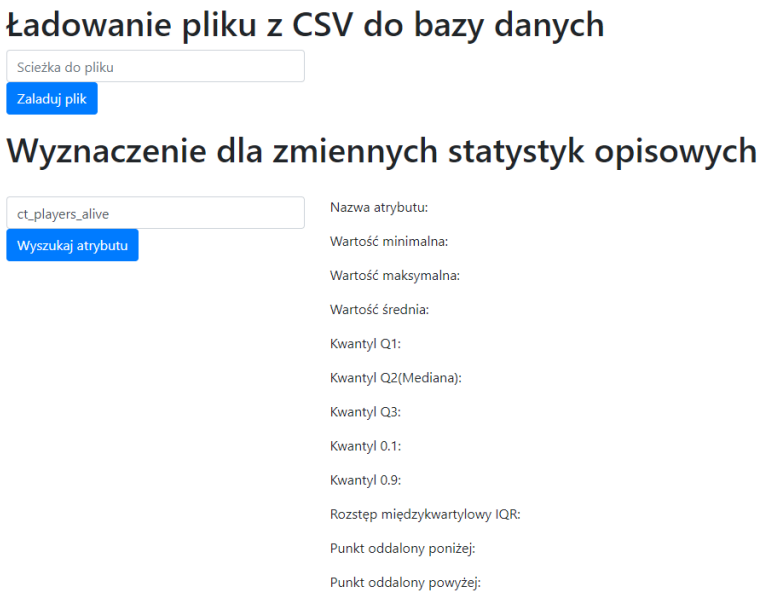
Założeniem projektu była możliwość przeprowadzenie procesu analizy statystycznej dla wybranego zestawu danych. W tym celu powstała aplikacja pracująca zgodnie z architekturą klient-serwer z wykorzystaniem implementacji architektury REST. W projekcie przewidziano poniższe funkcjonalności:

- Importowanie danych z pliku CSV do bazy danych
- Wyznaczanie zmiennych statystycznych takich jak:
    - Wartość minimalna, maksymalna, średnia
    - Kwantyl rzędu Q1, Q2, Q3
    - Kwantyl 0.1 i 0.9
    - Rozstęp międzykwartylowy IQR
    - Punkt oddalony poniżej i punkt oddalony powyżej
- Wyznaczenie współczynnika korelacji liniowej Pearsona dla 6 argumentów
- Wyznaczenie prostej regresji liniowej pomiędzy wybranymi parami zmiennych
- Wizualizacja danych w postaci histogramu oraz wykresu kołowego

Związku z wykorzystaniem architektury klient-serwer powstał interfejs graficzny wspomagany frameworkiem vue.js, który umożliwia komunikację z warstwą sprzętową. W celu przeprowadzenia obliczeń niezbędnych do wykonania założeń wykorzystano silnik bazodanowy MySQL i zapytania SQL, które znaczący sposób przyśpieszyły pracę. Dzięki wykorzystaniu funkcji matematycznych ryzyko błędu w błędnej implementacji znacząco spadło.

Pierwsza sekcja na stronie głównej umożliwia załadowania bazy danych danymi z pliku CSV. Działa ona na zasadzie odczytu i parsowania danych CSV do modelu danych, a następnie z modelu danych umieszczania rekordów w bazie danych z poziomu repozytoriów JPA.

Druga sekcja pozwala wyznaczać dla określonego atrybutu wartości statystyk opisowych takich jak o Wartość minimalna, maksymalna, średnia, kwantyl rzędu Q1, Q2, Q3, kwantyl 0.1 i 0.9, rozstęp międzykwartylowy IQR i punkt oddalony poniżej i punkt oddalony powyżej.



Rysunek 1.1 Interfejs graficzny sekcja pierwsza i druga, Źródło: Opracowanie własne

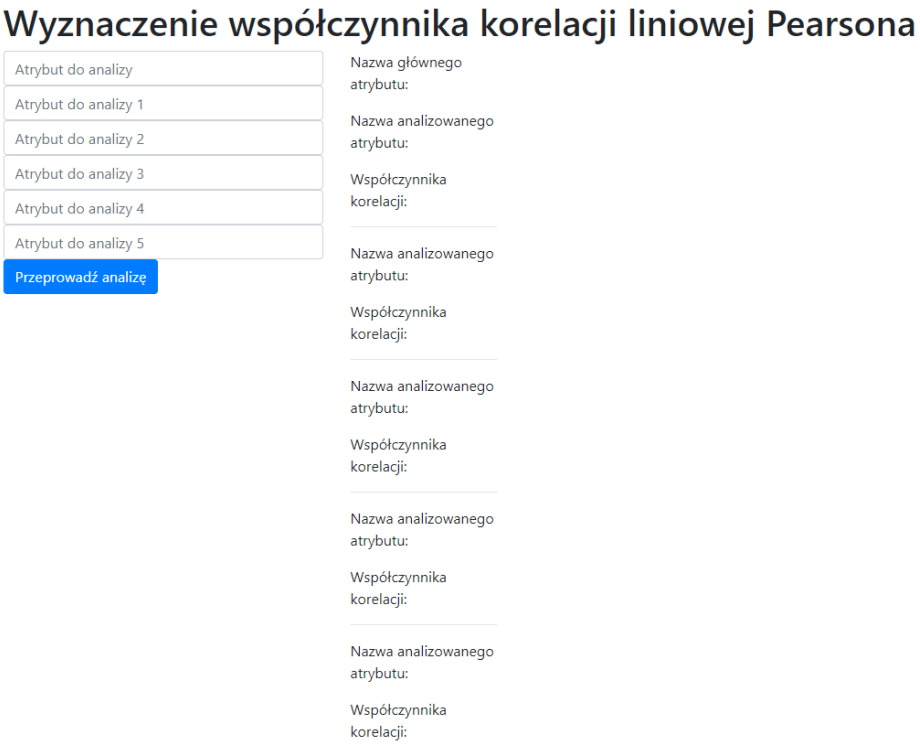
Dosyć ważnym elementem w procesie implementacji było obliczenie wartości dla odpowiedniego kwantylu. W tym celu wykorzystano zapytanie SQL jako natywne zapytanie z poziomu repozytorium JPA.

```
/**
Repozytorium JPA
*/
@Query(value = "SELECT * FROM (SELECT t.*,@row_num \\:=@row_num+1 AS
row_num FROM DataModel t,(SELECT @row_num \\:=0)counter ORDER BY :args)
temp WHERE temp.row_num=ROUND (:percentile* @row_num)
nativeQuery = true)
    DataModel getPercentile(@Param("percentile") double
percentile,@Param("args") String args);
```

Rysunek 1.2 Listing kodu obliczającego wartość dowolnego kwantylu, Źródło: Opracowanie własne

Zapytanie SQL przyjmuje dwa argumenty pierwszym o oznaczeniu „percentile” zawiera informacje jaki kwantyl ma być obliczony (w skali 0.1 do 0.9), drugim argumentem jest parametr „args”, który przekazuje wartość argumentu dla którego ma zostać obliczony kwantyl.

Trzecia sekcja posiada funkcjonalność wyznaczenia współczynnika korelacji liniowej Pearsona dla 6 atrybutów (główny atrybut i 5 opcjonalnych) przy czym współczynnik jest obliczany dla każdego z nich z osobna. W dostępne pola wprowadzamy nazwy atrybutów jakie nas interesują i następnie naciskamy na przycisk „przeprowadź analizę”.



Rysunek 1.3Interfejs graficzny sekcja trzecia, Źródło: Opracowanie własne

```
    @GetMapping(value = "/data/pcc/{attribute}/{attribute2}", produces =
"application/json")
    @ResponseBody
    public ResponseEntity<PearsonModel> getPearson(@Valid @PathVariable
String attribute, @Valid @PathVariable String attribute2) {
        Query sumXQuery = entityManager.createQuery("SELECT sum(" +
attribute + ") FROM DataModel");
        Query powerXQuery = entityManager.createNativeQuery("SELECT
sum(power(" + attribute + ",2)) FROM DataModel");
        Query sumYQuery = entityManager.createQuery("SELECT sum(" +
attribute2 + ") FROM DataModel");
        Query powerYQuery = entityManager.createNativeQuery("SELECT
sum(power(" + attribute2 + ",2)) FROM DataModel");
        Query sumXYQuery = entityManager.createQuery("SELECT sum(" +
attribute + "*" + attribute2 + ") FROM DataModel");
        Query countRecords = entityManager.createQuery("SELECT count(id)
FROM DataModel");

        double sumXValue =
Double.parseDouble(sumXQuery.getResultList().get(0).toString());
        double powerXValue =
Double.parseDouble(powerXQuery.getResultList().get(0).toString());
        double sumYValue =
Double.parseDouble(sumYQuery.getResultList().get(0).toString());
```

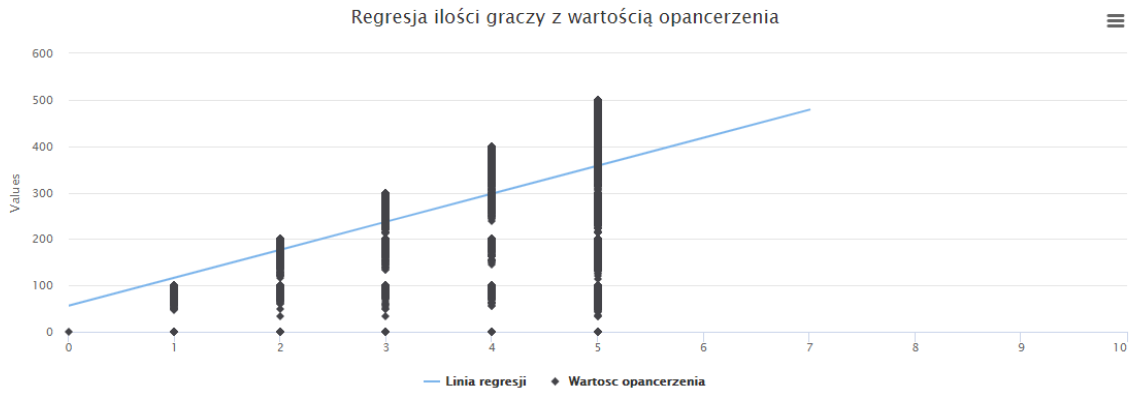
```
        double powerYValue =
Double.parseDouble(powerYQuery.getResultList().get(0).toString());
        double sumXYValue =
Double.parseDouble(sumXYQuery.getResultList().get(0).toString());
        double countRecordsValue =
Double.parseDouble(countRecords.getResultList().get(0).toString());
        double numeratorValue = (countRecordsValue * sumXYValue) -
(sumXValue * sumYValue);
        double denominatorValue = Math.sqrt(((countRecordsValue *
powerXValue) - Math.pow(sumXValue, 2)) * ((countRecordsValue *
powerYValue) - Math.pow(sumYValue, 2)));
        double pearsonValue = roundAvoid(numeratorValue /
denominatorValue, 2);
        PearsonModel pearsonModel = new PearsonModel(attribute,
attribute2, pearsonValue, sumXValue, powerXValue, sumYValue, powerYValue,
sumXYValue, countRecordsValue);
        return new ResponseEntity<>(pearsonModel, HttpStatus.OK);
    }
```

Rysunek 1.3 Listing kodu obliczającego współczynnik korelacji liniowej Pearsona, Źródło: Opracowanie własne

Powyższy kod dostępny jest z przestrzeni adresowej /data/pcc podając dwa atrybuty, główny i dla którego ma został obliczona korelacja. Następnie poprzez zapytania SQL pobierane są dane na temat sumy atrybutów, sumy pierwiastka atrybutów a także ilości rekordów. Następnie zgodnie ze wzorem na obliczenie korelacji liniowej Pearsona (Wzór pobrany z literatury o oznaczeniu 5) jest obliczany współczynnik i zwracany jako model danych PearsonModel, który zawiera takie dane jak nazwa atrybutu pierwszego i drugiego, a także wszystkie wartości obliczone w celu wyznaczenia współczynnika korelacji liniowej Pearsona.

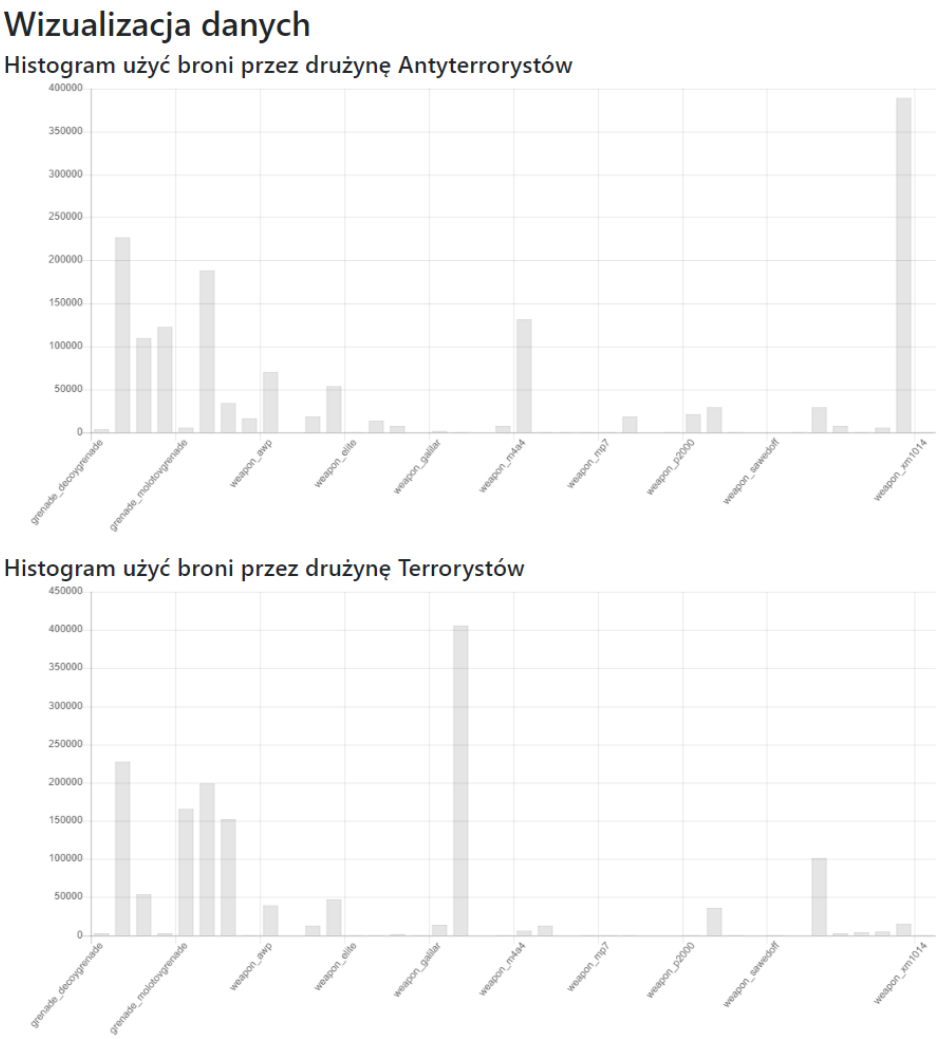
Czwarta sekcja umożliwia obliczenie prostej regresji liniowej pomiędzy wybranymi parami zmiennych. Podobnie jak w poprzedniej sekcji w tym celu wykorzystano ResponseEntity, który umożliwi wykonanie kodu Java po wejściu na adres /data/linearregression/ i podaniu potrzebnych atrybutów. W celu obliczenia regresji liniowej ponownie wykorzystano zapytania SQL i zestaw funkcji matematycznych w celu obliczenia regresji liniowej (Wzór pobrany z literatury o oznaczeniu 6). Jest także dostępna wizualizacja w formie wykresu regresji liniowej.

Rysunek 1.4 Interfejs graficzny sekcja czwarta, Źródło: Opracowanie własne



Rysunek 1.5 Wykres regresji liniowej, Źródło: Opracowanie własne

W ostatniej sekcji możemy zwizualizować takie dane jak histogramy użyć broni dla obydwu drużyn lub która drużyna odniosła więcej zwycięstw. Dane do wykresów są pobierane z bazy danych i generowane pliki JSON. Dzięki takiemu rozwiązaniu wykresy są generowane w czasie rzeczywistym.



Rysunek 1.6 Wizualizacja danych, Źródło: Opracowanie własne

Wyniki testowania procesu analizy danych

Testowanie systemu polegało na wykorzystaniu wszystkich elementów systemu dla głównych atrybutów.

Testowanie zmiennych statystycznych dla atrybutów podanych w tabeli:

Nazwa argumentu	time_left	ct_score	t_score	ct_health	t_health
Wartość minimalna:	0.01	0	0	0	0
Wartość maksymalna:	175	32	33	500	500
Wartość średnia:	97.89	6.71	6.78	412.11	402.71
Kwantyl Q1:	54.92	3	3	350	322
Kwantyl Q2(Mediana):	94.91	6	6	500	500
Kwantyl Q3:	166.92	10	10	500	500
Kwantyl 0.1:	27.75	1	1	198	169
Kwantyl 0.9:	174.95	13	14	500	500
Rozstęp międzykwartylowy IQR:	111.99	7	7	150	178
Punkt oddalony poniżej:	53.42	1.5	1.5	348.5	320.5
Punkt oddalony powyżej:	168.42	11.5	11.5	501.5	501.5

Nazwa argumentu	ct_armor	t_armor	ct_money	ct_health	round_winner
Wartość minimalna:	0	0	0	0	0
Wartość maksymalna:	500	500	80000	80000	1
Wartość średnia:	314.14	298.44	9789.02	11241.04	0.51
Kwantyl Q1:	194	174	1300	1550	0
Kwantyl Q2(Mediana):	377	334	5500	7150	1
Kwantyl Q3:	486	468	14600	18000	1
Kwantyl 0.1:	0	0	450	550	0
Kwantyl 0.9:	500	500	24800	27850	1
Rozstęp międzykwartylowy IQR:	292	294	13300	16450	1
Punkt oddalony poniżej:	192.5	172.5	1298.5	1548.5	-1.5
Punkt oddalony powyżej:	487.5	469.5	14601.5	18001.5	2.5

Testowanie wyznaczenia współczynnika korelacji liniowej Pearsona dla argumentów:

Nazwa argumentu	time_left	ct_money	t_money	ct_health	t_health
time_left	1	0.37	0.34	0.68	0.68
ct_money	0.37	1	0.37	0.31	0.24
t_money	0.34	0.37	1	0.22	0.31
ct_health	0.68	0.31	0.22	1	0.76
t_health	0.68	0.24	0.31	0.76	1

Nazwa argumentu	ct_armor	map	bomb_planted	ct_players_alive	t_players_alive
ct_armor	1	0	-0.28	0.43	0.15
map	0	1	-0.03	0	-0.01
bomb_planted	-0.28	-0.03	1	-0.62	-.04
ct_players_alive	0.43	0	-0.62	1	0.63
t_players_alive	0.15	-0.01	-0.4	0.63	1

Testowanie wyznaczenia regresji liniowej dla par zmiennych:

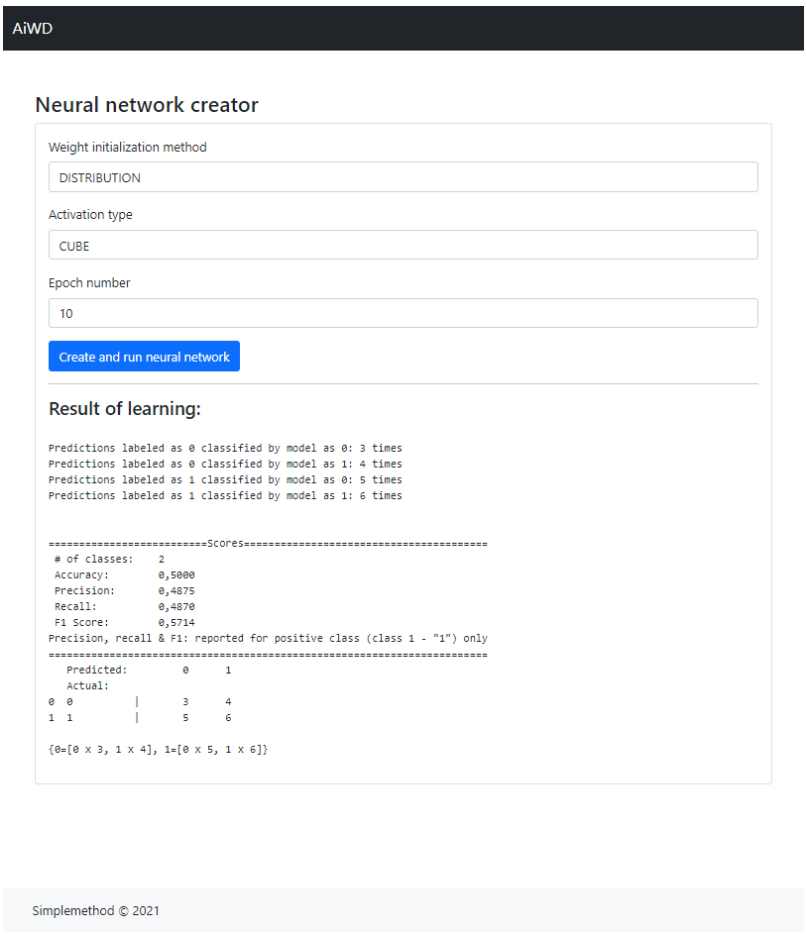
Nazwa argumentu głównego	Nazwa argumentu drugiego	Wartość atrybutu głównego	Przewidywana wartość argumentu drugiego	Formuła regresji
ct_players_alive	ct_armor	2	176.876	y=56.13578003127992+60.36998215879007*x
ct_players_alive	ct_armor	5	357.986	y=56.13578003127992+60.36998215879007*x
ct_players_alive	t_armor	2	255.085	y=216.94627805799905+19.06951754357323*x
ct_players_alive	t_armor	5	312.294	y=216.94627805799905+19.06951754357323*x
ct_players_alive	t_players_alive	2	2.817	y=1.543109512747653+0.6371632697019207*x
ct_players_alive	t_players_alive	5	4.729	y=1.543109512747653+0.6371632697019207*x
t_players_alive	t_armor	2	174.449	y=65.0181493374231+54.71548672524824*x
t_players_alive	t_armor	5	338.596	y=65.0181493374231+54.71548672524824*x

Wnioski

Jak widać z powyższych danych wszystkie elementy systemu funkcjonują poprawnie najlepiej można to zaobserwować przy wyliczaniu korelacji liniowej Pearsona gdzie dane są z przedziału od -1 do 1 i dla tych samych argumentów wartość zawsze wynosi 1 co jest potwierdzeniem poprawnej implementacji tego rozwiązania. Zastosowanie regresji liniowej umożliwia nam hipotetyczne określenie wartości dla danego argumentu funkcji. Dzięki obliczeniom kwantyli możemy określić w której części jest największe skupisko danych.

Projekt procesu analizy eksploracyjnej i implementacja

Założeniem projektu była możliwość przeprowadzenie procesu analizy eksploracyjnej dla wybranego zestawu danych. W tym celu powstała aplikacja pracująca zgodnie z architekturą klient-serwer z wykorzystaniem implementacji architektury REST. W projekcie przewidziano i zaimplementowano takie testowania sieci neuronowych dla jednej z wybranych funkcji inicjalizacji wag, określonego typu aktywacji neuronu, a także ilości epok. Sieć neuronowa ma na celu klasyfikację która drużyna wygra na podstawie danych takich jak: Czy bomba została podłożona, ilość życia poszczególnych drużyn i czas do końca rundy. Jako informacje zwrotną zostaje wygenerowany komunikat w którym możemy dowiedzieć się o dokładności etapu uczenia, precyzji danych, poziom ufności modelu, a także jego ocenę ewaluacji podczas testów. Komunikat zwraca także tablicę pomyłek w formie tabeli i tablicę pomyłek w formie zapisu macierzowego. Przy przygotowaniu aplikacji wykorzystano takie same rozwiązania i technologie jak przy aplikacji do implementacji analizy statystycznej. Nową biblioteką było wykorzystanie deeplearning4j służącej do opracowania sieci neuronowej. Dane testowe i uczące są przygotowane w jednym pliku csv zaimportowanym bezpośrednio w strukturę aplikacji.



Rysunek 1.7 Interfejs graficzny aplikacji przeznaczonej do eksploracji danych, Źródło: Opracowanie własne

```
  try (RecordReader recordReader = new CSVRecordReader(0, ',')) {
            recordReader.initialize(new FileSplit(
                    new ClassPathResource("csgo_full.csv").getFile()
            ));
            DataSetIterator iterator = new
RecordReaderDataSetIterator(recordReader, 50, 4, 2);
            DataSet allData = iterator.next();
            allData.shuffle(20);
            return dataNormalization(allData);
        } catch (Exception e) {
            Thread.dumpStack();
            new Exception("Stack trace").printStackTrace();
            System.out.println("Error: " + e.getLocalizedMessage());
        }
        return null;
```

Rysunek 1.8 Listing przetwarzający plik CVS do formatu DataSet, Źródło: Opracowanie własne

Powyższy kod ma na celu załadowanie pliku csv o nazwie csgo_full, a następnie poddać go procesowi dopasowania poprzez stworzenie iteratora i przeszukiwanie pliku wiersz po wierszu z uwzględnieniem danych zawartych w RecordReaderDataSetIterator, czyli przeszukiwaniu tylko czterech kolumn i uwzględnieniu ostatniej jako kolumny z podziałem na klasy w tym przypadku 2, albo wygrywa strona terrorystów, albo antyterrorystów. Powyższy kod odwołuje się także do metody dataNormalization, który kod został zaprezentowany poniżej.

```
private DataSet dataNormalization(DataSet dataSet) {
        NormalizerMinMaxScaler normalizer = new NormalizerMinMaxScaler(-
1.0, 1.0);
        normalizer.fit(dataSet);
        normalizer.transform(dataSet);
        return dataSet;
    }
```

Rysunek 1.9 Listing kodu z normalizacją danych z wykorzystaniem NormalizerMinMaxScaler, Źródło: Opracowanie własne

Wyżej zaprezentowany kod umożliwia normalizację danych z zakresu od -1 do 1 z wykorzystaniem mechanizmu NormalizerMinMaxScaler. Rozwiązuje to problem ręcznego i mozolnego procesu normalizacji danych i ogranicza ryzyko błędnego znormalizowania danych.

```
public String makeNNetwork(WeightInit weightInit, Activation activation,
DataSet normalizerData, Integer epoch) {
        SplitTestAndTrain testAndTrain =
normalizerData.splitTestAndTrain(0.65);
        DataSet trainingData = testAndTrain.getTrain();
        DataSet testingData = testAndTrain.getTest();

        MultiLayerConfiguration configuration = new
NeuralNetConfiguration.Builder()
                .weightInit(weightInit)
                .activation(activation)
                .updater(new Nesterovs(0.9, 0.9))
                .l2(0.001)
                .list()
                .layer(0, new
DenseLayer.Builder().nIn(4).nOut(7).build())
```

```
                .layer(1, new
DenseLayer.Builder().activation(Activation.SOFTMAX).nIn(7).nOut(5).build(
))
                .layer(2, new
DenseLayer.Builder().activation(Activation.SOFTMAX).nIn(5).nOut(3).build(
))
                .layer(3, new
OutputLayer.Builder(LossFunctions.LossFunction.MSE).activation(Activation
.SOFTMAX).nIn(3).nOut(2).build())
                .backprop(true).pretrain(false)
                .build();

        MultiLayerNetwork model = new MultiLayerNetwork(configuration);
        model.init();
        for (int i = 0; i < epoch; i++) {
            model.fit(trainingData);
        }
        INDArray output = model.output(testingData.getFeatureMatrix());
        Evaluation eval = new Evaluation(2);
        eval.eval(testingData.getLabels(), output);
        String s = eval.stats() + "\r\n" + eval.confusionToString() +
"\r\n" + eval.getConfusionMatrix();
        return s;
    }
```

Rysunek 1.10 Listing kodu z procesem tworzenia sieci neuronowej, Źródło: Opracowanie własne

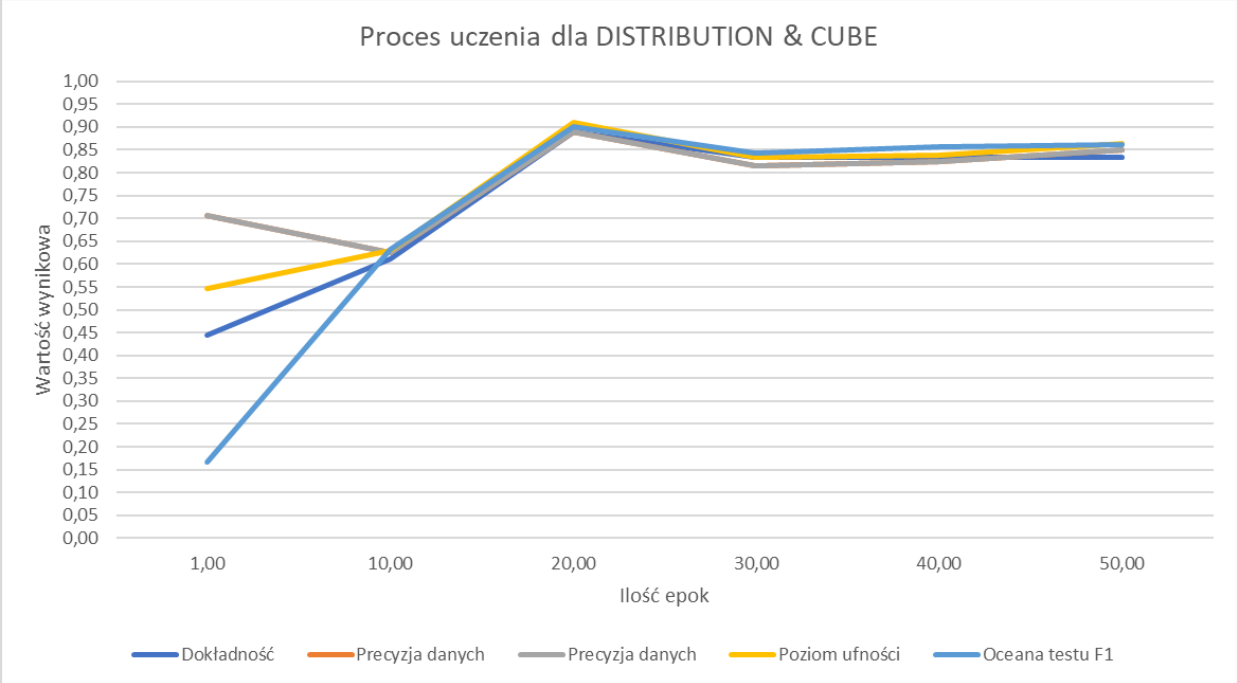
Powyższy listing kodu przedstawia proces tworzenia sieci neuronowej. Metoda przyjmuje trzy argumenty, funkcję inicjalizacji wag, metodę aktywacji neuronów oraz ilość epok. Sama sieć składa się z dwóch warstw ukrytych w pierwszej mamy 7 neuronów aktywowanych funkcją SOFTMAX, druga warstwa ukryta posiada ich 5 także aktywowanych funkcją SOFTMAX. Dodatkowo sieć neuronowa została wyposażona w mechanizm regularyzacji L2 o wartości 0.001. Warstwa wyjściowa sieci zgodnie z ilością klas posiada tylko dwa neurony wyjściowe. Metoda SplitTestAndTrain rozdziela zbiór danych w stosunku 65% dane uczące i 35% dane testowe. Po procesie uczenia sieci następuje jego ewaluacja poprzez wykorzystanie klasy Evaluation i na końcu działania metody jest tworzony ciąg znaków String z danymi wynikowymi. Komunikat zwraca także tablicę pomyłek w formie tabeli i tablicę pomyłek w formie zapisu macierzowego – W celu implementacji wyników została wykorzystana wiedza z literatury 1,2,3,4

Wyniki testowania procesu eksploracji danych

Testowanie procesu eksploracji danych polegało na przetestowaniu czterech zestawów funkcji inicjalizacji wag początkowych oraz funkcji aktywacji w epokach 1,10,20,30,40,50, a następnie zapisanie wyników i zaprezentowanie ich w formie tabeli oraz wykresu.

Testowanie zestawu inicjalizacja wag: Rozkład dystrybuanty, funkcja aktywacji Cube

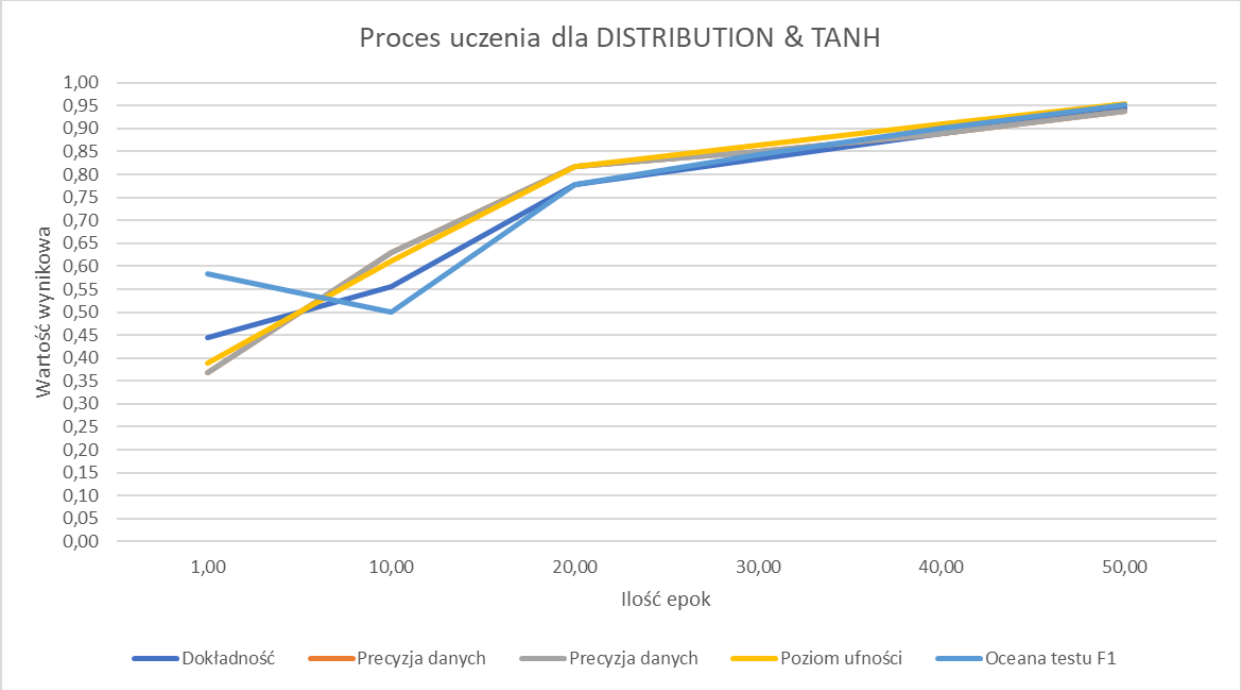
Proces uczenia	Ilość epok	Dokładność	Precyzja danych	Poziom ufności	Ocena testu F1
DISTRIBUTION & CUBE	1,00	0,44	0,71	0,55	0,17
DISTRIBUTION & CUBE	10,00	0,61	0,63	0,63	0,63
DISTRIBUTION & CUBE	20,00	0,89	0,89	0,91	0,90
DISTRIBUTION & CUBE	30,00	0,83	0,82	0,83	0,84
DISTRIBUTION & CUBE	40,00	0,83	0,83	0,84	0,86
DISTRIBUTION & CUBE	50,00	0,83	0,85	0,86	0,86



Powyższy zestaw najlepiej zrealizował swój proces dla 20 epok gdzie w teście F1 osiągnął 0,90, natomiast dla pozostałych epok 30,40,50 ten wynik znacząco spadł (z 0,9 do 0,86 dla epok 40 i 50). Może to wynikać z niezwykle szczęśliwego rozkładu danych i sposobu inicjalizacji wag.

Testowanie zestawu inicjalizacja wag: Rozkład dystrybuanty, funkcja aktywacji Tanh

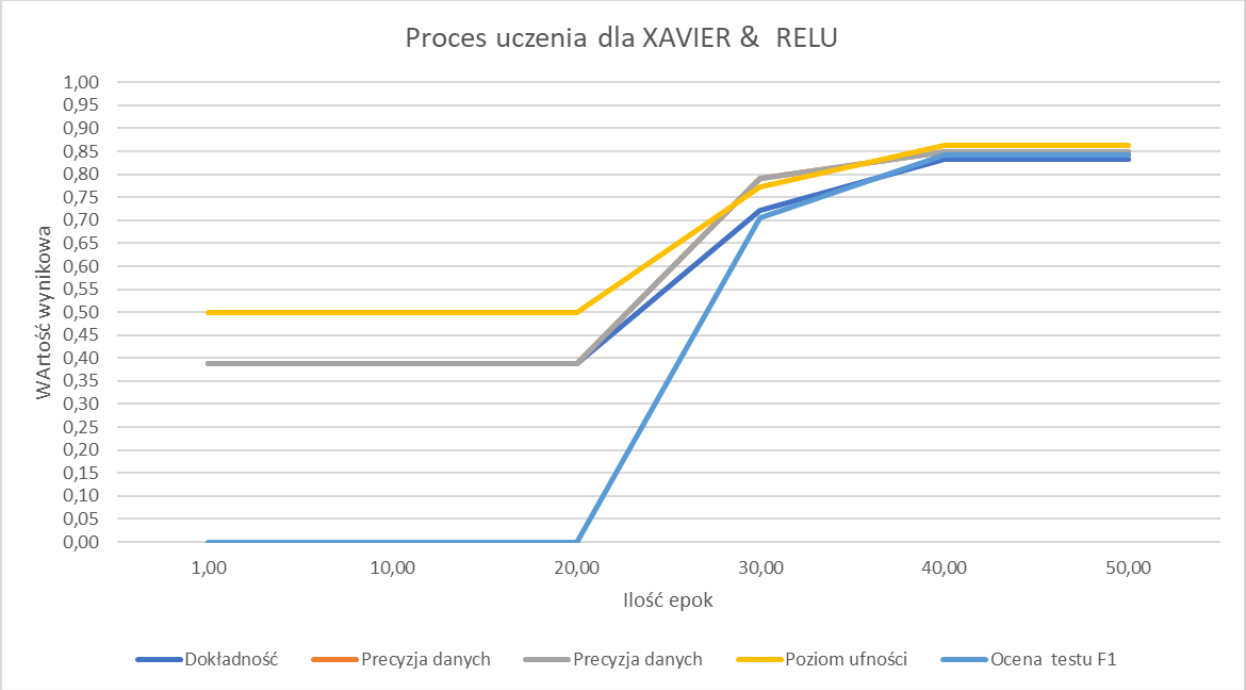
Proces uczenia	Ilość epok	Dokładność	Precyzja danych	Poziom ufności	Ocena testu F1
DISTRIBUTION & TANH	1,00	0,44	0,37	0,39	0,58
DISTRIBUTION & TANH	10,00	0,56	0,63	0,61	0,50
DISTRIBUTION & TANH	20,00	0,78	0,82	0,82	0,78
DISTRIBUTION & TANH	30,00	0,83	0,85	0,86	0,84
DISTRIBUTION & TANH	40,00	0,89	0,89	0,91	0,90
DISTRIBUTION & TANH	50,00	0,94	0,94	0,95	0,95



W tym zestawie danych odnotowano spadek oceny testu systemu F1 w 10 epoce po czym można zaobserwować wzrost tych danych aż do końca badanego obszaru ilości epok. Warto zauważyć, że dla tego zestawu danych model osiągnął poziom dokładności, precyzji i ufności aż do poziomu 0.95 co jest niezwykle wysokim wynikiem.

Testowanie zestawu inicjalizacja wag: XAVIER, funkcja aktywacji RELU

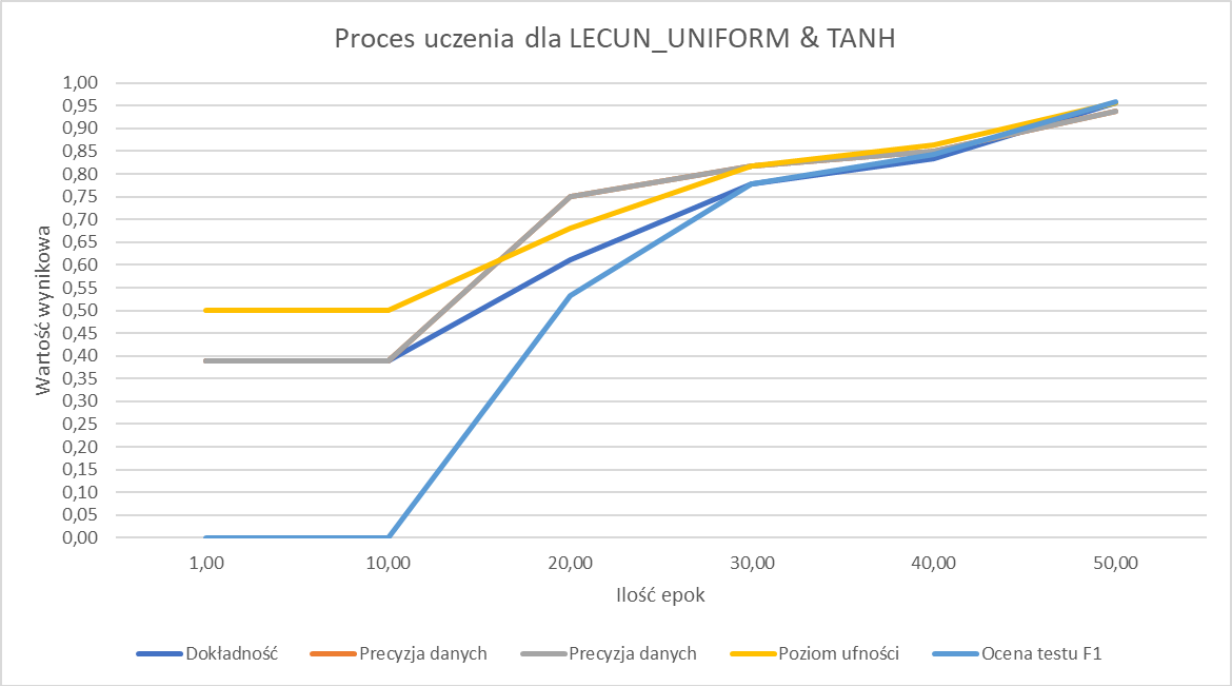
Proces uczenia	Ilość epok	Dokładność	Precyzja danych	Poziom ufności	Ocena testu F1
XAVIER & RELU	1,00	0,39	0,39	0,50	0,00
XAVIER & RELU	10,00	0,39	0,39	0,50	0,00
XAVIER & RELU	20,00	0,39	0,39	0,50	0,00
XAVIER & RELU	30,00	0,72	0,79	0,77	0,71
XAVIER & RELU	40,00	0,83	0,85	0,86	0,84
XAVIER & RELU	50,00	0,83	0,85	0,86	0,84



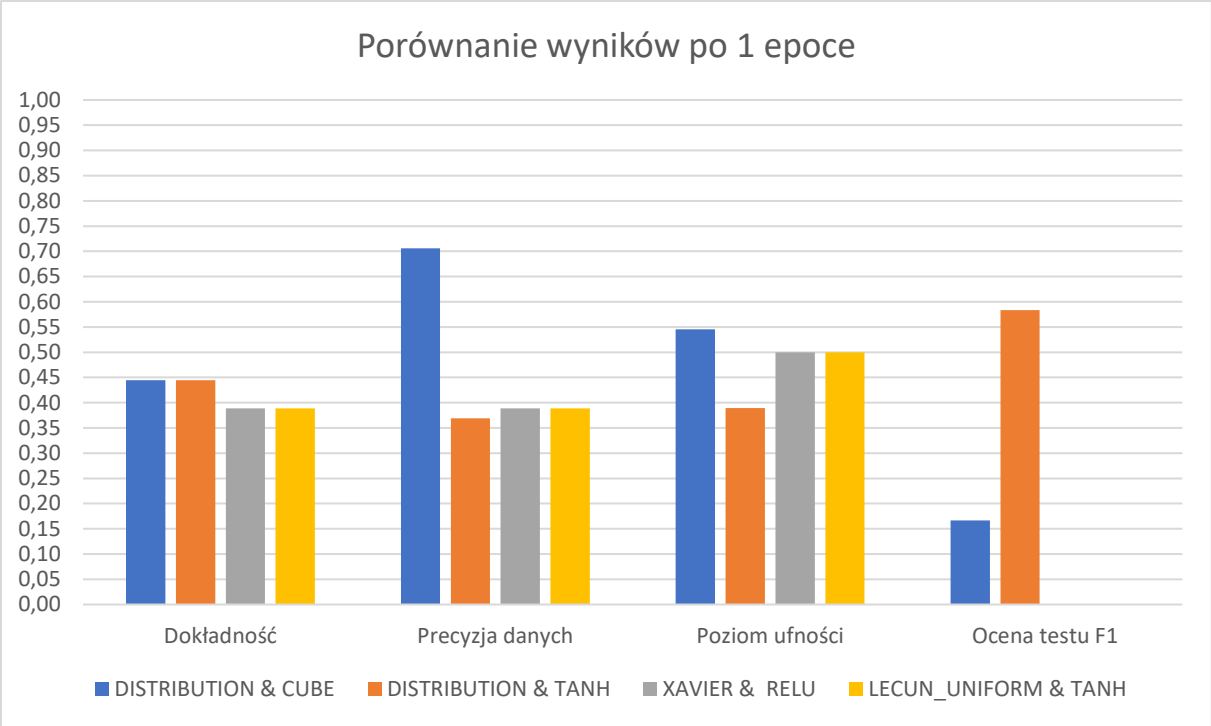
Ten zestawie danych dla epok 1,10 i 20 w ogóle nie odnotował procesu uczenia osiągając 0,00 w teście F1 co całkowicie dyskwalifikuje to rozwiązanie dla takiej ilości epok. Dopiero od epoki 30 odnotowano wzrost tych danych aż do końca badanego obszaru ilości epok. Warto zauważyć, że dla epok 40 i 50 odnotowano takie same wartości co może sugerować, że został osiągnięty maksimum wartości uczenia dla tej konfiguracji.

Testowanie zestawu inicjalizacja wag: LECUN_UNIFORM, funkcja aktywacji TANH

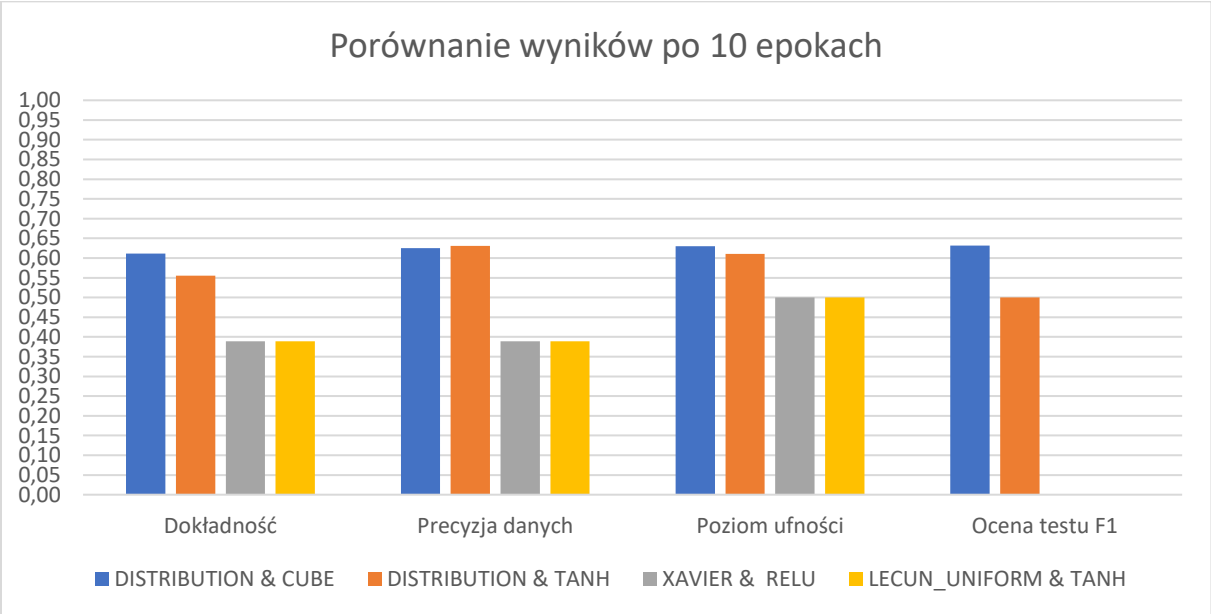
Proces uczenia	Ilość epok	Dokładność	Precyzja danych	Poziom ufności	Ocena testu F1
LECUN_UNIFORM & TANH	1,00	0,39	0,39	0,50	0,00
LECUN_UNIFORM & TANH	10,00	0,39	0,39	0,50	0,00
LECUN_UNIFORM & TANH	20,00	0,61	0,75	0,68	0,53
LECUN_UNIFORM & TANH	30,00	0,78	0,82	0,82	0,78
LECUN_UNIFORM & TANH	40,00	0,83	0,85	0,86	0,84
LECUN_UNIFORM & TANH	50,00	0,96	0,94	0,96	0,96



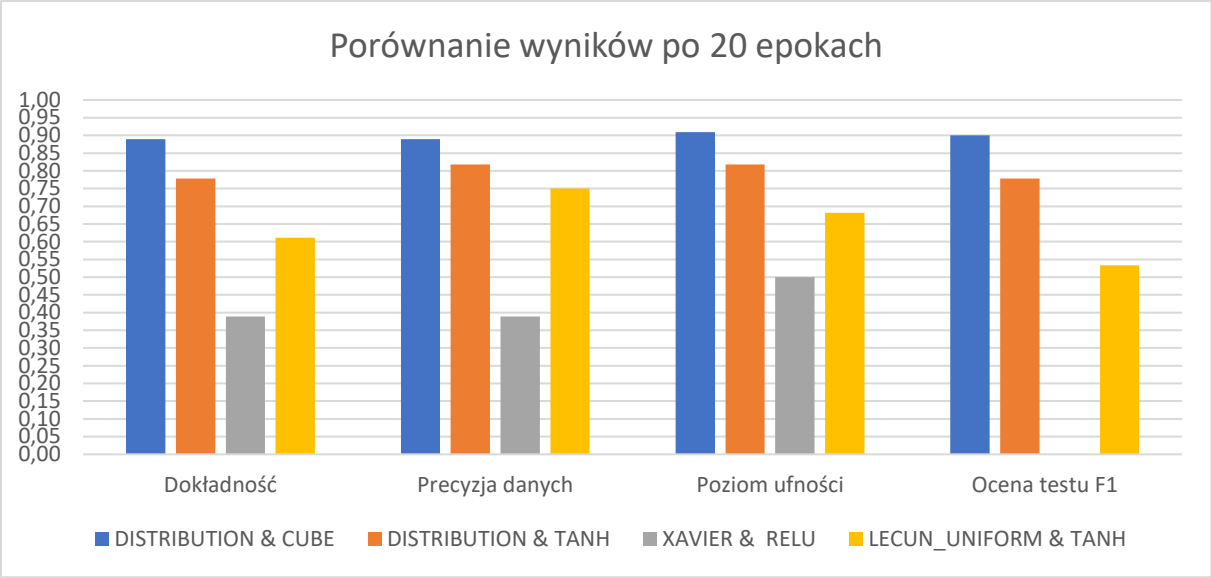
Ten zestawie danych podobnie jak w poprzedni dla epok 1 oraz 10 w ogóle nie odnotował procesu uczenia osiągając 0,00 w teście F1 co całkowicie dyskwalifikuje to rozwiązanie dla takiej ilości epok. Dopiero od epoki 20 odnotowano wzrost tych danych aż do końca badanego obszaru ilości epok. Dla epoki 50 udało się osiągnąć wartości 0,96 dla większości argumentów w tym testu F1.



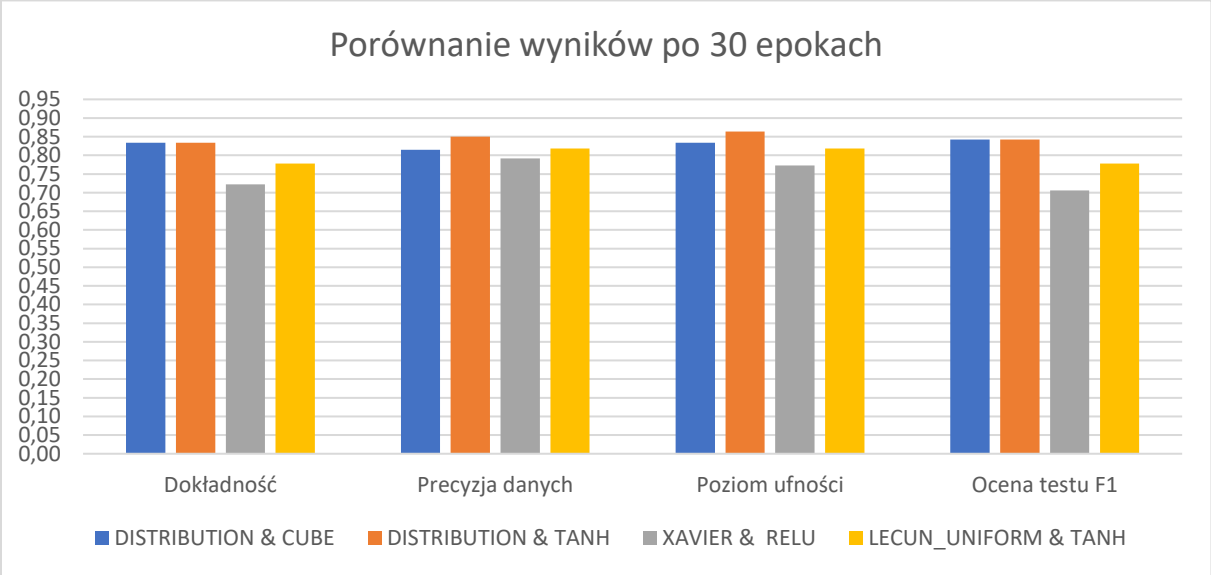
Po pierwszej epoce widać, ze zestaw DISTRIBUTION & CUBE najlepiej spisał się w pierwszej epoce znacząco przewyższając pozostałe modele w precyzji danych natomiast przegrywając w ocenie testu F1.



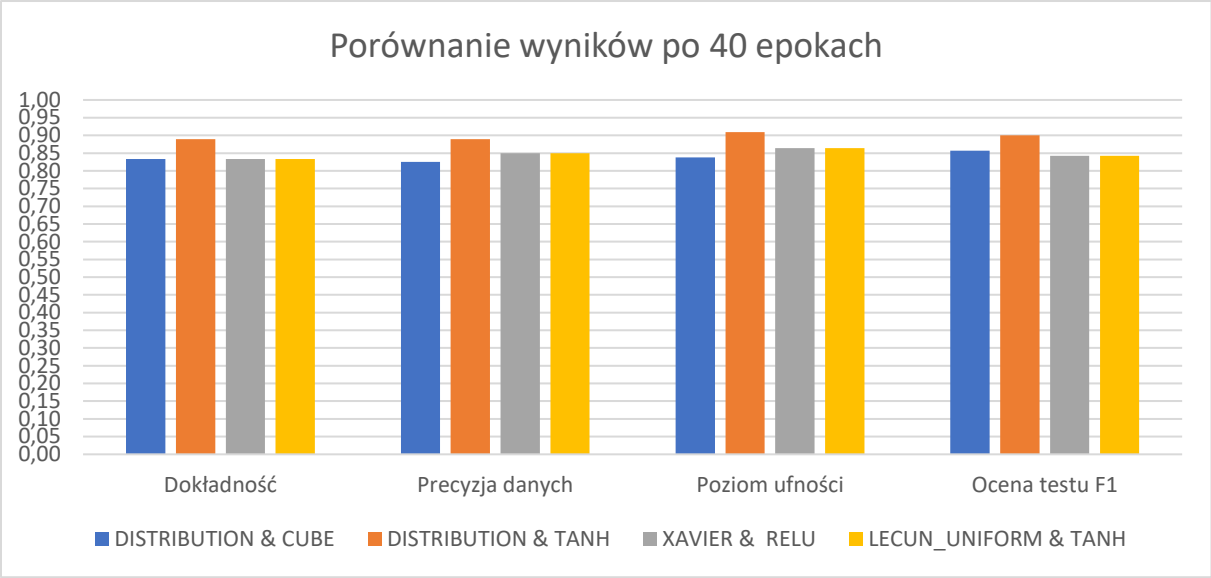
Po dziesiątej epoce można było zauważyć nadal przewagę DISTRIBUTION & CUBE we wszystkich badanych parametrach oprócz precyzji danych gdzie na prowadzenie wyszedł zestaw DISTRIBUTION & TANH. Pozostałe dwa zestawy uzyskały wynik testu F1 na poziomie 0,00.



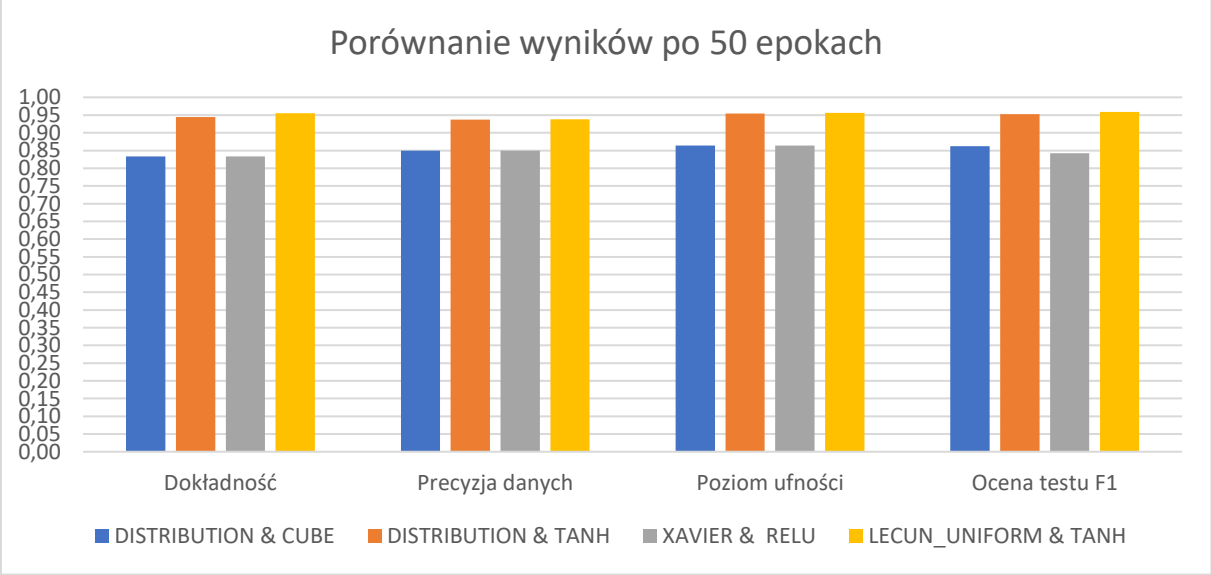
20 epoka uczenia to powiększenie przewagi zestawu DISTRIBUTION & CUBE nad innymi osiągając wartości bliskie 0,90 wynika to najprawdopodobniej z niezwykle szczęśliwego rozkładu danych. Na drugim miejscu tak jak poprzednio pozostaje DISTRIBUTION & TANH. Zestaw XAVIER & RELU jako jedyny nie zaliczył testu F1 ponownie osiągając wynik 0,00.



W 30 epoce można zauważyć, że DISTRIBUTION & TANH wysunął się na prowadzenie i osiągnął jako jedyny poziom ufności przekraczając 0,85. Wszystkie zestawy algorytmów zaliczyły już testy F1.



40 epoka można zauważyć stopniowy wzrost wyników dla XAVIER & RELU i LECUN_UNIFORM & TANH. Widać, że przewaga DISTRIBUTION & TANH znacząco zmalała od poprzedniej epoki, dodatkowo można zauważyć, ze DISTRIBUTION & CUBE osiągnął swój maksymalny poziom nauczenia i wynik jego się nie zmienia znacząco.



50 epoka była znaczącym zaskoczeniem, na minimalne prowadzenie przesunął się zestaw LECUN_UNIFORM & TANH osiągając 0,96 w teście F1, w poziomie ufności danych i dokładności oraz 0,94 w precyzji danych. Drugi zestaw DISTRIBUTION & TANH osiągnął odpowiednio 0,95 w teście F1 oraz w poziomie ufności, a także 0,94 w dokładności i precyzji danych.

Wnioski

Jak widać z powyższych danych wszystkie elementy systemu funkcjonują poprawnie najlepiej można to zaobserwować analizując przebieg procesu uczenia dla jednych z zestawu danych. Najlepszym zestawem okazał się LECUN_UNIFORM & TANH osiągając w teście F1 0,96 co jest niezwykle wysokim wynikiem. Dosyć dużym zaskoczeniem był zestaw DISTRIBUTION & CUBE, który pomimo początkowych obiecujących wyników w połowie testu przestał dominować i ostatecznie osiągnął w teście F1 0,86 co plasuje go na przedostatnim miejscu.

Podsumowanie

W ramach pracy zostało opracowane i zaimplementowane spełniające wszystkie wymagania, którego głównym celem było stworzenie rozwiązania przeznaczonego do analizy statystycznej i eksploracyjnej. Dzięki wykorzystaniu frameworku Spring Boot można było stworzyć proste API służące do komunikacji interfejsu graficznego z warstwą serwerową. Wykorzystanie deeplearning4j pozwoliło nie tylko na stworzenie sieci neuronowej, ale także na szybką i bezpieczną normalizację danych. Sam zbiór danych także okazał się być niezwykle przyjazny dla użytkownika i procesu analizy statystycznej oraz eksploracyjnej.

Wykaz literatury

1. [https://www.youtube.com/watch?v=aircAruvnKk] Dostęp z dnia: 01.01.2021 - cała seria
2. [https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall] Dostęp z dnia: 10.01.2021
3. [https://developers.google.com/machine-learning/crash-course/classification/accuracy] Dostęp z dnia: 10.01.2021
4. [https://developers.google.com/machine-learning/crash-course/classification/true-false-positive-negative] Dostęp z dnia: 10.01.2021
5. [https://www.youtube.com/watch?v=2-VEwB8n7Fo] Dostęp z dnia: 10.01.2021
6. [https://youtu.be/AZborBAKNjs] Dostęp z dnia: 10.01.2021