

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH
TEHNOLOGIJA OSIJEK

Računarstvo usluga i analiza podataka

SEMINARSKI RAD

“Predviđanje ishoda borbe pokemona”

Matej Šarčević, Daria Čaćić

Osijek, 2021.

SADRŽAJ

1. UVOD	1
2. TIJEK RJEŠAVANJA ZADATKA.....	2
2.1. Opis korištenih podataka	2
2.2. Model strojnog učenja.....	3
2.3. Web aplikacija.....	5
2.4. Grafički prikaz.....	8
2.5. Baza podataka.....	9
3. RJEŠENJE.....	10
4. ZAKLJUČAK	13
5. POVEZNICE I LITERATURA	14

1. UVOD

Pokemon je puno toga. Od tvrtke, video igara, igraćih karata, mange, animea, raznih aplikacija i još mnogo toga, zbog čega je teško naći osobu koja nije čula za taj pojam. Početci Pokemona su bili 1995. godine, prije preko 25 godina, kada je japanac Satoshi Tajiri kreirao Pokemon svijet. Sljedeće godine u drugom mjesecu izašla je prva RPG igra za Game boy "Pokemon Red Version" i "Pokemon Blue Version" (u japanu originalno kao "Pocket Monster Red and Green"). Nakon nekoliko mjeseci počele su izlaziti i igraće karte, a 1997. počela je izlaziti manga temeljena na Pokemon svijetu koju je pisao Hidenori Kusaka. Iste godine počinje izlaziti i anime, danas vrlo poznat, što govori činjenica da se prikazuje u 169 država diljem svijeta. Tijekom sljedećih godina kreirana je tvrtka, izašli su razni filmovi, igre, a 2009. održalo se i prvo svjetsko natjecanje. Prošlost Pokemona je vrlo opsežna zbog raznih postignuća, zbog čega bi bilo previše ulaziti u daljnje opise. S obzirom na popularnost, mogu se naći i mnoge baze podataka koje se odnose na Pokemone poput korištene u ovom projektnom zadatku. S obzirom na opseg i točnost podataka moguće je iskoristiti ih za razne svrhe, a tako i za strojno učenje. Cilj ovog projekta je napraviti praktični simulator borbe dvaju pokemona temeljen na uzetim podacima iz baza (koje će biti opisane kasnije) i pomoću programskog jezika Pythona napraviti model strojnog učenja, koristiti framework u istom jeziku za kreiranje web sučelja, uređivanje web stranice i postavljanje aplikacije na cloud platformu.

2. TIJEK RJEŠAVANJA ZADATKA

Zbog većeg broja različitih koraka u rješavanju zadatka, ovaj dio je podijeljen na više sekcija gdje je svaka posvećena specifičnom zadatku.

2.1. Opis korištenih podataka

Za izradu modela potrebno je naći prikladne podatke za učenje. Pošto se radi o borbi pokemona ti podatci moraju biti potrebne informacije o pokemonima koje utječu na ishod borbi i neki oblik zapisa borbi na temelju kojih se može izvući zaključak o pobjedniku. Korišteni podatci su 3 csv datoteke (url za stranicu nalazi se u dijelu seminara “*poveznice i literatura*”). Pokemon.csv je dataset koji sadrži 12 stupaca sa atributima: Id, Name, Type 1, Type 2, Health, Attack, Defense, Special Attack, Special Defense, Speed, Generation, Legendary. U drugom dataset-u combats.csv nalaze se 3 stupca: Id, Id i Winner gdje je Id poveznica sa prijašnjim datasetom, a Winner pobjednik dvoboja. Ta dva seta podataka korištena su za treniranje modela. Neki od podataka:

(pokemon.csv):

5, Charmander, Fire, , 39, 52, 43, 60, 50, 65, 1, False
9, Mega Charizard Y, Fire, Flying, 78, 104, 78, 159, 115, 100, 1, False
187, Pichu, Electric, , 20, 40, 15, 35, 35, 60, 2, False

Može se vidjeti da atribut Type 2 u pokemon.csv može biti prazan ukoliko je nepostojeći za tog Pokemona.

(combats.csv):

694, 747, 694
455, 549, 549

(tests.csv):

129, 117
660, 211

Da bi se mreža istrenirala na tim podacima potrebno je nekako povezati dva skupa podatka (pokemon.csv i combats.csv). Nakon toga bi se na temelju razlike između određenih atributa istrenirala mreža. Iz tog razloga za predikciju pobjede pokemona potrebno je imati sve attribute prisutne za tijekom treniranja mreže. U ovu svrhu unutar

programa kreirana je funkcija koja će pripremiti podatke za treniranje i kasnije za predikciju.

```
def get_stats_dif(pokemons_df):
    pokemon_stats =
    ["HP", "Attack", "Defense", "Sp_Atk", "Sp_Def", "Speed", "Generation", "Legendary"]
    stats_df = pokemon[pokemon_stats].T.to_dict("list")
    first_pokemon = pokemons_df.First_pokemon.map(stats_df)
    second_pokemon = pokemons_df.Second_pokemon.map(stats_df)
    new_list = []
    for i in range(len(first_pokemon)):
        new_list.append(np.array(first_pokemon[i]) -
np.array(second_pokemon[i]))
    new_pokemon_df = pd.DataFrame(new_list, columns = pokemon_stats)
    return new_pokemon_df
```

2.1.1 Funkcija za pripremu podataka

Dataset tests.csv koristit će se samo za testiranje mreže.

2.2. Model strojnog učenja

Određivanje pobjednika između dva pokemona svodi se na problem klasifikacije i stoga će se pri treniranju modela koristiti algoritmi najprikladniji za rješavanje ovog zadatka. Kako bi se optimiziralo predviđanje najbolje je testirati nekoliko modela za klasifikaciju. A ti modeli su: linearna regresija, k najbližih susjeda, random forest algoritam, gradient boosting algoritam i support vector clustering algoritam. Algoritam koji pokazuje najbolju točnost uzet će se kao najbolji. Prije testiranja algoritama poziva se već spomenuta funkcija *get_stats_dif* i nakon toga potrebno je podijeliti podatke na test i train. U ovom slučaju omjer je 7:3. Također potrebno je podijeliti x i y skupove tj. na podatke na temelju kojih predviđa i varijablu koju predviđa.

```

data = get_stats_dif(combats)
data = pd.concat([data, combats.Winner], axis = 1)

x_label = data.drop("Winner", axis = 1)
y_label = data["Winner"]

x_train, x_test, y_train, y_test = train_test_split(x_label, y_label,
test_size = 0.3)

```

2.2.1 Korištenje nove funkcije, podjela na x/y i test/train

Već spomenuti algoritmi uzeti su iz razloga što su među najboljima za binarnu klasifikaciju što je slučaj ovdje. Kod njihovim implementacija i treniranja prikazan je na slici ispod.

```

from sklearn.linear_model import LogisticRegression
Log_Reg = LogisticRegression()
model_1 = Log_Reg.fit(x_train, y_train)
prediction = model_1.predict(x_test)
print('Accuracy LR: ', accuracy_score(prediction, y_test) * 100)

from sklearn.neighbors import KNeighborsClassifier
KNN = KNeighborsClassifier(n_neighbors = 5)
model_2 = KNN.fit(x_train, y_train)
prediction = model_2.predict(x_test)
print('Accuracy KNN: ', accuracy_score(prediction, y_test) * 100)

from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier
Rand_Forest = RandomForestClassifier(n_estimators = 100,
min_samples_split = 2, max_features =
"auto", class_weight=("balanced_subsample"))
model_3 = Rand_Forest.fit(x_train, y_train)
prediction = model_3.predict(x_test)
print('Accuracy RFC: ', accuracy_score(prediction, y_test) * 100)

Grad_Boosting = GradientBoostingClassifier( n_estimators = 100,
max_depth = 1, learning_rate = 1, random_state = 0)
model_4 = Grad_Boosting.fit(x_train, y_train)
prediction = model_4.predict(x_test)
print('Accuracy GB: ', accuracy_score(prediction, y_test) * 100)

from sklearn.svm import NuSVC
Svc = NuSVC()
model_5 = Svc.fit(x_train, y_train)
prediction = model_5.predict(x_test)
print('Accuracy SVC: ', accuracy_score(prediction, y_test) * 100)

```

2.2.2 Implementacija klasifikacijskih modela u kodu

Rezultat koji ispisuje ovaj kod dan je na slici ispod.

```
Accuracy LR: 88.88000000000001
Accuracy KNN: 90.30666666666667
Accuracy RFC: 94.75333333333333
Accuracy GB: 94.08666666666666
Accuracy SVC: 89.02666666666667
```

2.2.3 Ispis rezultata

Iz rezultata se vidi da najveću točnost daje random forest algoritam, a gradient boosting je vrlo blizu njega. Ostali su neusporedivo lošiji sa 5 ili više postotka manje od najboljeg rezultata. Sveukupno svi daju relativno dobre rezultate iz čega se može zaključiti da su podaci uzeti za testiranje poprilično precizni i korisni u određivanju pobjednika u pokemon dvoboju. S obzirom na rezultate izabran je random forest algoritam koji daje najbolju točnost od 95%. On je poprilično precizan, brz i relativno lagan za koristiti zbog čega je jedan od najčešće korištenih algoritama za strojno učenje.

Prije pisanja koda za framework, potrebno je exportati model kako bi se mogao koristiti. U tu svrhu se koristi nova biblioteka *joblib*. Funkcija *dump* kreira datoteku koja koja će se kasnije pozivati pomoću funkcije *load*.

```
import joblib
joblib.dump(model_3, 'modeljoblib')
```

2.2.4 Kreiranje modela

2.3. Web aplikacija

Kao framework u sklopu ovog projekta korišten je Flask. Flask je Python web framework koji se koristi za razvoj web aplikacija. Temelji se na Werkzeug WSGI alatu i Jinja2. Werkzeug je WSGI poslužitelj zadužen za rute i debugiranje. Jinja2 je zadužena za podršku rada s predlošcima. Osim ove dvije komponente, Flask se sastoji i od Click komponente, koja služi za olakšan rad u komandnom načinu rada. Zbog svoje veličine smatra se micro-frameworkom, što ga čini popularnim i jednostavnim za učenje web programiranja u Pythonu. Iako je micro-framework, Flask je dizajniran tako da bude lako proširiv uz pomoć Flask ekstenzija, tako da uz sve osnovne mogućnosti koje nudi, pomoću ekstenzija se može proširiti ovisno o tome što je

potrebno na web aplikaciji koja se izrađuje (npr autentifikacija, web forme, baze podataka itd.)

Flask se instalira pomoću pip-a, koji je alat za upravljanje i instalaciju Python paketa s PyPI repozitorija. Komanda koja se koristi za instalaciju je: `pip install flask`. Prije kreiranja funkcija za web stranicu potrebno je kreirati instancu klase Flask i konfigurirati datoteke koje se koriste za dohvaćanje slika, spremanje grafova i kao baza podataka čije funkcionalnosti će se kasnije spomenuti.

```
app = Flask(__name__)
app.config['upload_icons_folder']='icons'
app.config['upload_folder']='plots'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///previous_battles.db'
```

2.3.1 Kreiranje instance Flask i konfiguriranje datoteka

Za prikaz web stranice se koristi nekoliko html datoteka koje predstavljaju zasebne povezane stranice. Njihov izgled je osmišljen u zasebnoj css datoteci. Detaljnije informacije o svakoj stranici posebno bit će komentirane u poglavlju “*rješenje*” zajedno sa slikovnim primjerima. Projekt sveukupno sadrži 4 html dokumenta koji predstavljaju 4 stranice. Početka ima svrhu unošenja imena pokemona i može voditi na sve ostale 3 stranice. U python datoteci početna je stranice povezana pomoću funkcije *home()* koja vraća html te stranice.

```
@app.route('/')
def home():
    return render_template('home.html')
```

2.3.2 Funkcija home za učitavanje početne stranice

Pri unosu pokemona stranica nudi i prijedloge pokemona preko elementa datalist u koji su uneseni svi pokemoni iz baze podataka koja je korištena za učenje modela.


```
<datalist id="pokemons">
  <option value="Bulbasaur">
  <option value="Ivysaur">
  <option value="Venusaur">
  <option value="Mega Venusaur">
  <option value="Charmander">
  <option value="Charmeleon">
  <option value="Charizard">
  <option value="Mega Charizard X">
  <option value="Mega Charizard Y">
```

2.3.3 Datalist koji sadrži sve pokemone iz csv datoteke

Glavna funkcionalnost ove web stranice nalazi se u funkciji *post()* koja se aktivira odmah pri klikom na button ili pritiskom tipke *enter* i vodi do glavne html datoteke. Unutar te funkcije odmah se poziva već spomenuta funkcija *load* za učitavanje modela.

```
def post():
    model =
    joblib.load('modeljoblib')
```

2.3.4 Učitavanje modela u aplikaciju

Nakon toga se vrši provjera unesenih podataka. Ukoliko se krivo unesu pokemoni ili nema ulaza početna stranica vodi do stranice koja obavještava o grešci. Ako su točno uneseni onda se podatci spremaju u data frame koji je prvi korak u pripremanju podataka za unos u model. Nakon toga se pretvaraju u oblik korišten za predikcijski model pomoću već spomenute funkcije *get_stats_dif* i radi se predikcija pomoću učitanoog modela.

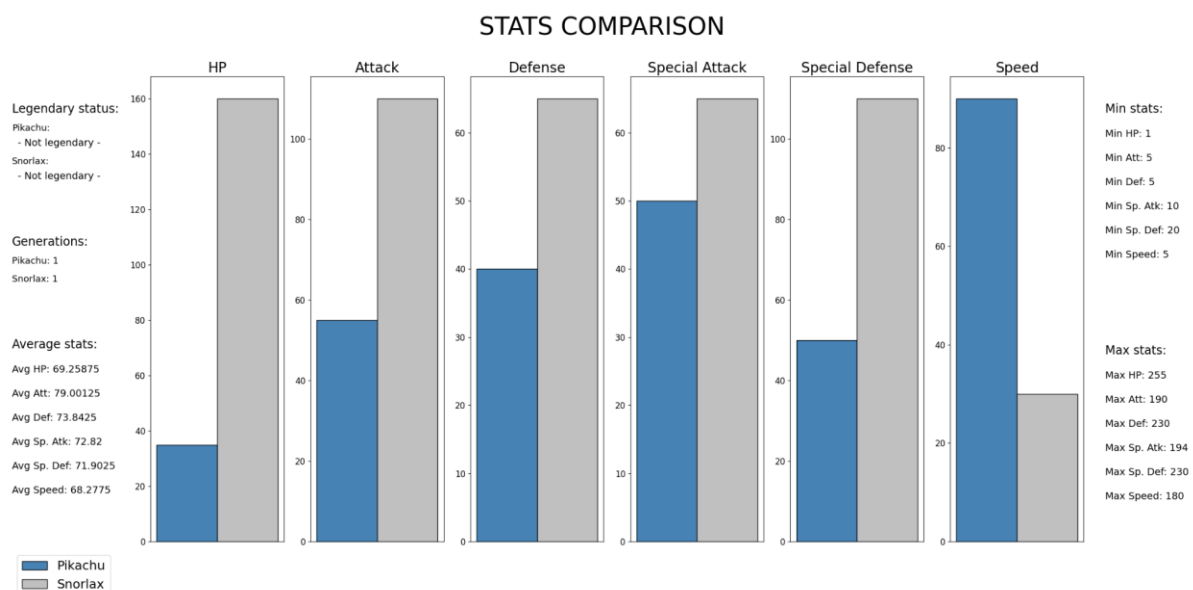
```
new_pokemon_df = get_stats_dif(pokemon_df)
prediction = model.predict(new_pokemon_df)
```

2.3.5 Određivanje pobjednika unutar aplikacije

Pošto je za ovu stranicu potrebno više podataka za prikaz u ovoj funkciji računaju se i ostale stvari poput vjerojatnosti, kreiranje grafa, dohvaćanje slika i spremanje u bazu koji će se kasnije detaljnije opisati. Nakon što je sve to izračunato učitava se stranica za prikaz rezultata. Osim na već spomenute dvije stranice, početna može voditi i na treću u kojoj se nalazi prikaz prijašnjih bitki kojima se pristupa iz baze podataka.

2.4. Grafički prikaz

Radi boljeg razumijevanja rezultata dobro je uključiti i grafički prikaz uz njihovu objavu. To će pomoću u shvaćanju kako na rezultat utječu atributi pokemona te pomoći u budućih borbama. Ovdje je prikaz riješen pomoću biblioteke `matplotlib` unutar `python` datoteke. Pomoću stupčastog grafa se uspoređuju većina stats-a pokemona uz tekstualne podatke po strani poput `legendary status-a` i generacije koje nema smisla grafički prikazivati. Također dodani su podatci o prosječnim, minimalnim i maksimalnim stats-ima (koji su prikazani u grafu) pokemona kako bi se moglo usporediti njihov individualan status jačine ovisno o tim vrijednostima. Rezultantni grafički prikaz je prikazan na slici ispod. Ovaj prikaz se također sprema kao slika u datoteku koja je konfigurirana na početku aplikacije.



2.5.1 Grafički prikaz stats-a sa svim dodatnim informacijama

Za kreiranje stats comparison zbog veće količine informacija sukladno tome ima dosta koda, ali se taj kod ovdje neće prikazivati kako se ne bi previše udaljavallo od glavne zadaće projekta. Cijeli kod se može naći na github stranici čiji link će biti spomenut u dijelu *poveznice i literatura*.

2.5. Baza podataka

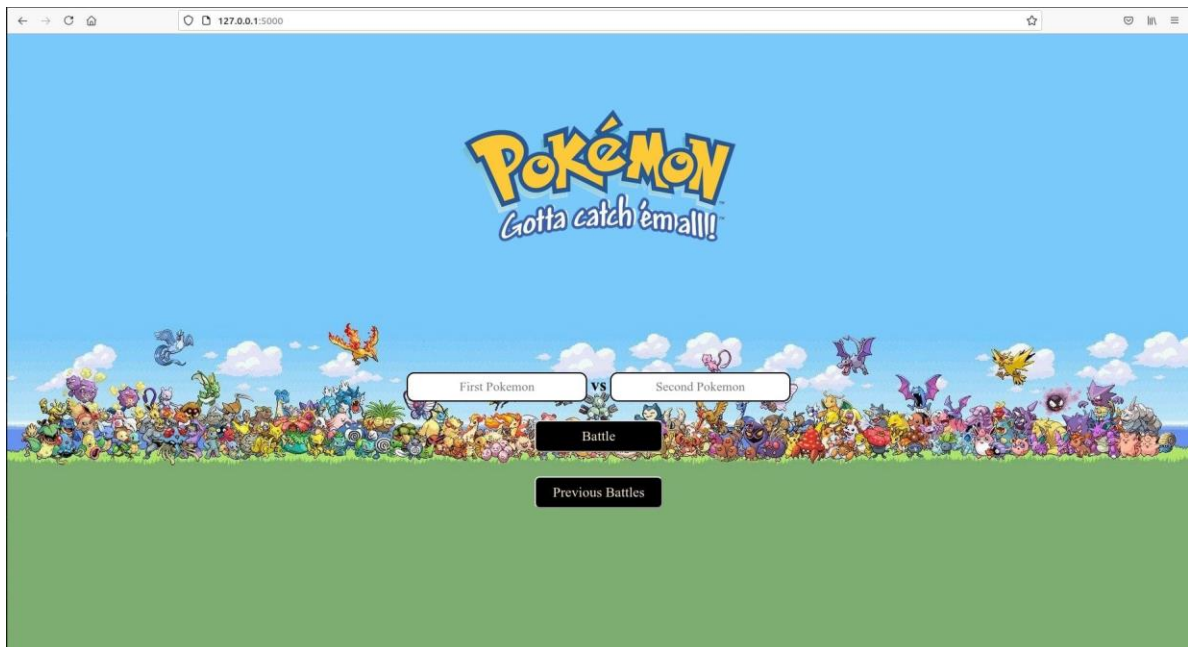
Za spremanje informacija o prethodnim borbama pokemona u ovom projektu korišten je SQLite. SQLite je relacijska baza podataka temeljena na maloj C programskoj biblioteci. Baš zbog svoje jednostavnosti je izabrana navedena baza podataka. Za stvaranje i povezivanje sa web aplikacijom potrebno je kreirati datoteku (npr. "database.db"), te nakon toga u u projekt uvesti modul sqlite3 i napraviti konekciju na datoteku baze podataka. Osim samog SQLite-a, korišten je i ORM SQLAlchemy za povezivanje podataka sa bazom podataka.

```
SQLite version 3.31.1 2020-01-27 19:55:54
Enter ".help" for usage hints.
sqlite> .schema
CREATE TABLE previous_battles (
    "ID" INTEGER NOT NULL,
    "firstPokemon" VARCHAR(50) NOT NULL,
    "firstPokemonID" INTEGER NOT NULL,
    "firstImagePath" VARCHAR(50) NOT NULL,
    "secondPokemon" VARCHAR(50) NOT NULL,
    "secondPokemonID" INTEGER NOT NULL,
    "secondImagePath" VARCHAR(50) NOT NULL,
    winner VARCHAR(50) NOT NULL,
    PRIMARY KEY ("ID")
);
```

2.6.1 Baza podataka

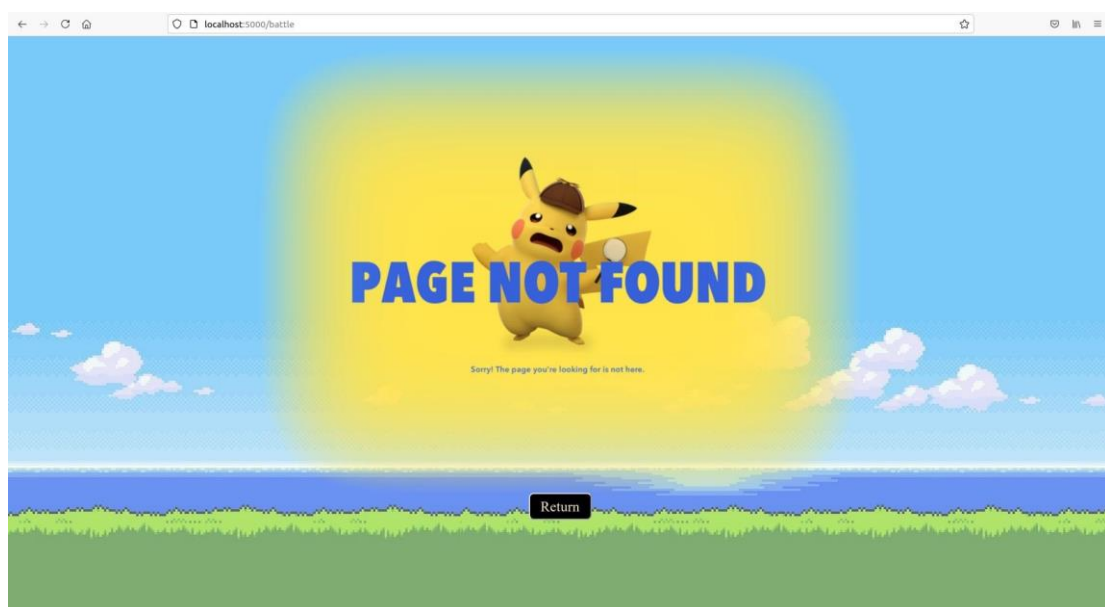
3. RJEŠENJE

Već je spomenuto postoje 4 html dokumenta i ukratko opisane njihove uloge. Ovdje će se malo detaljnije opisati njihov sadržaj uz slikovne primjere. Početka stranica sadržava dva polja za input u koje se unose pokemoni za koje se hoće predvidjeti pobjednik, gumb za realiziranje borbe i gumb za pristup prijašnjim borbama.



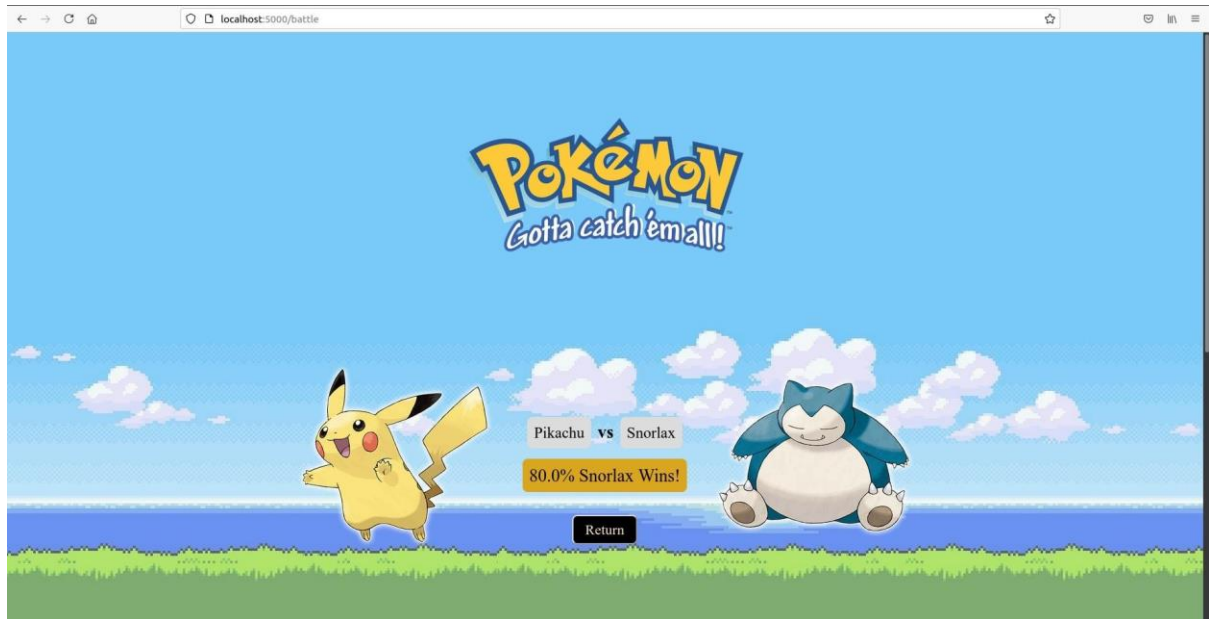
3.1. Početna stranica

Klikom na gumb “*Battle*” ili pritiskom tipke *enter* ukoliko nisu točno uneseni podatci i ako se ti podatci ne nalaze u bazi dolazi se na sljedeću html stranicu.



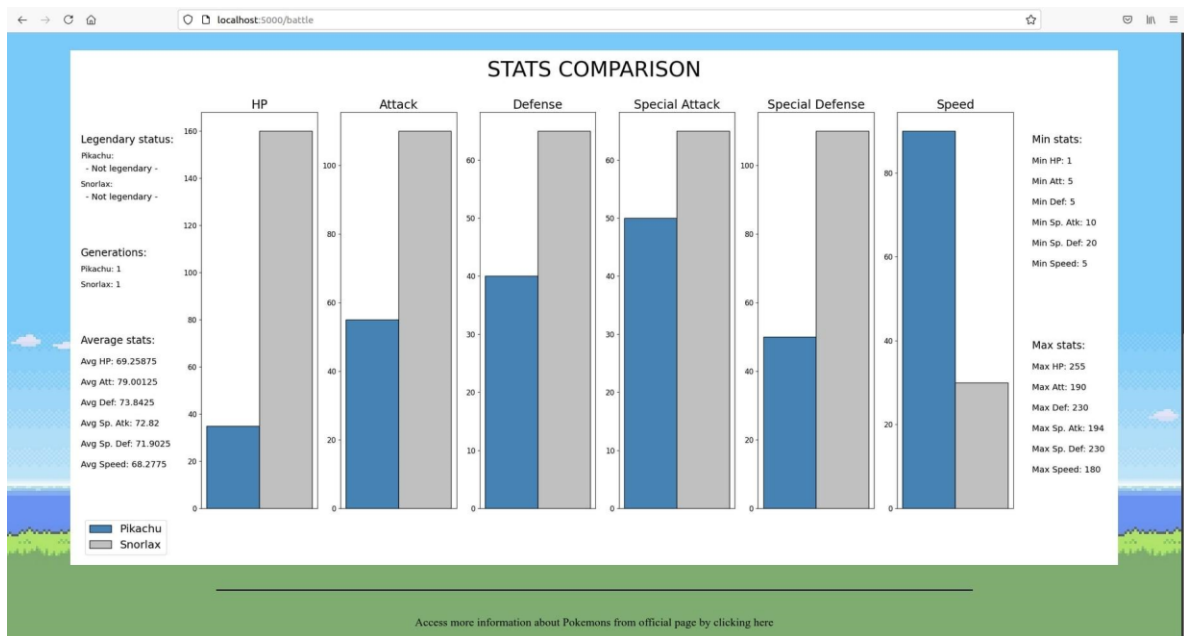
3.2. Stranica za netočan unos

Ova stranica nema nikakve funkcionalnosti osim povratka na početnu. Njena svrha je samo obavijestiti o neuspjehu izračuna rezultata. Ukoliko je unos točan i postoji vrše se sva potrebna računanja unutar spomenute *post()* funkcija i otvara se stranica za prikaz rezultata.















3.3 Stranica za prikaz rezultata

Na njoj su prikazani pokemoni sa svojim slikama koje su skinute sa originalne pokemon stranice i izvlače se iz datoteke u kojoj se te slike nalaze pomoću funkcije *picture(x)*. Osim toga objavljen je pobjednik i postotak vjerojatnosti da pobjedi uz grafički prikaz usporedbe njihovih stats-a. Funkcionalnosti ove stranice su gumb koji vodi nazad na početnu stranicu.



3.4 Grafički prikaz stat-sa pokemona iz borbe

Nakon povratka na početnu stranicu može se pristupiti podacima o prijašnjim pobjednicima klikom na gumb “*Previous Battles*”. Na ovoj stranici se nalazi tablični prikaz pokemona, njihovih ikona i pobjednika dvoboja. Maksimalan broj podataka je XX koji kada se prekorači izbacuje najstariji podatak kako bi se prikazao noviji.

First Pokemon ID	First Pokemon	First Pokemon Image	Second Pokemon ID	Second Pokemon	Second Pokemon Image	Winner
31	Pikachu		156	Snorlax		Snorlax
133	Scyther		137	Pinsir		Scyther
361	Vibrava		384	Castform		Castform
488	Mime Jr.		692	Mandibuzz		Mandibuzz
388	Mega Banette		140	Magikarp		Magikarp
141	Gyarados		166	Mew		Mew

Return

3.5 Stranica za tablični prikaz prijašnjih bitki

4. ZAKLJUČAK

Predviđanje ishoda borbi pokemona zanimljiv je i interaktivan način korištenja strojnog učenja. Iznenadujuće, za ovaj problem nije bilo teško pronaći bazu podataka i postoje razne sa različitim pokemonima. Uzeta je najveća i najkvalitetnija za treniranje modela za koji je uzet random forest algoritam koji je iznimno dobar za ovaj tip klasifikacije i daje točnost od 95% što je vrlo dobar rezultat. Kako se ne bi radilo samo o pokretanju koda kreirana je i web aplikacija pomoću frameworka Flask. Aplikacija nudi ne samo predviđanje unesenih pobjednika nego grafički prikaz usporedbe njihovih stats-a te bazu podataka za praćenje prijašnjih borbi. Za sve to korišten je Python programski jezik što pokazuje njegovu moć i fleksibilnost, a i jednostavno korištenje za strojno učenje jer nudi mnoge biblioteke i gotove klase.

5. POVEZNICE I LITERATURA

https://sg.portal-pokemon.com/play/pokedex?fbclid=IwAR2Z3QGRpE46KWK8TJbK3TRctJNbGqsjEGo74_OJPjS405XnnhfSZQA9okQ

<https://builtin.com/data-science/random-forest-algorithm>

<https://corporate.pokemon.co.jp/en/aboutus/history/>

https://bulbapedia.bulbagarden.net/wiki/2009_World_Championships

<https://github.com/mattejs/PokemonBattle>