CS C3240 – Machine Learning Project

Prediction of match time in the game SMITE Battleground Of The Gods

Introduction

I will be focusing on the 1v1 gamemode in Smite called Duel. In duel two players battle against each other until one player defeats the enemy's base titan. The players choose a character from over 100 different choices and can then equip themselves with different items. I will try to determinate the match times for different matchups and player skills to see if the character choice is the biggest factor or if difference in the players' skills play a bigger role.

Problem Formulation

I will try to answer the question; given player A and B and their character choice, how long will the match go on before one of the players win?

The datapoints consist of match data and player data. The label is the match time, and the features are player level, player mmr (matchmaking rating) and difference in character choice. The player level shows how long and how much a player has played the game over the years, so general skill in the game. Mmr shows how good a player is in a specific gamemode, so in this case their duel mmr. To get the character choice as a numerical value I will be using the characters winrate in duel that I get from https://ranked.casualsmite.com/ and then I take the difference between the choices. The reason why I only take the difference between "stats" in the character choice is that the difference between a level 1 and 2 player is not the same as between a level 100 and 101 but the difference in character choice is always the same. All the features and the label are floating point numbers.

I will have to fetch and compile the data myself because it is not available to the public. To get the data I will be using Smite-API (https://webcdn.hirezstudios.com/hirez-studios/legal/smite-api-developer-guide.pdf). With this API I can collect match data and then I can access player data from the match data. Sadly, the API has a daily limit of requests which means that I might not get more than a couple thousand datapoints.

Methods

The dataset consists of about 2000 datapoints at the moment but I will collect more data over time. Every datapoint consists of player level 1 and 2, mmr 1 and 2, character difference and the match duration. As I already mentioned earlier all the features and the label are floating point numbers. To actually get the data I needed I had to clean up the input from the API and I also had to cross-reference the character choice with a table of win rates for every character and then I also needed to take the difference in the choices.

I already explained above what the different features are, but I chose them specifically for different reasons. I selected the player level because it shows the players general knowledge and skill of the game. The mmr was chosen because it is basically the same as the player level but for the specific gamemode duel. I also included the character choice difference because from my own experience of the game I know that some character matchups will always be heavily favored to one side and almost always end in a short match.

I have chosen polynomial regression as my ML model. This is because I do not think that linear regression is a fitting model for me. This is because I do not think that player skill is linear. In the beginning of the game the player learns a lot but the longer you play it gets harder to find new techniques and to improve your previous skills. I also tried both polynomial and linear regression and the polynomial on some degrees gave a smaller error than linear regression. As loss function, I chose mean squared error. This is because it was familiar from the course and it is easy to use with polynomial regression and the libraries that we have used.

I split the data into training, validation and testing sets using the train_test_split function. I split them so that the size of the training set is half of the whole dataset, the validation set is 30% of the whole dataset and lastly the testing is then 20% of the dataset. I chose these sizes because we used these values on the course earlier.

ML Code

March 10, 2022

[ ]:
```
import smite
import json
import csv

#
# This is the data collection part of the code
#
# I am using the annotation God instead of character because all the characters␣
 ↪in the game are gods from different mythologies
#

dict = {}

def findWinRate(godName):
    if godName in dict:
        return dict.get(godName)
    return 0

gods = open('gods.csv', 'r')
reader = csv.DictReader(gods)
for row in reader:
    dict[row['God']] = row['Rate']

file = open('res.csv', 'w', newline='')
writer = csv.writer(file)
writer.writerow(["AccountLevel1", "AccountLevel2", "MMR1", "MMR2", "God1",␣
 ↪"God2", "Duration"])

#dev_id = Cannot show becuase of legal reasons
#auth_key = Same on this one

client = smite.SmiteClient(dev_id, auth_key)
input_dict = client.get_match_ids_by_queue(440, 20220307)
matches = [x['Match'] for x in input_dict if x['Active_Flag'] == 'n']
for x in matches:
    matchInfo = client.get_match_details(x)
    filtered = [{"Account_Level": x['Account_Level'], "Duration":␣
↪x['Match_Duration'], "MMR" : x['Rank_Stat_Duel'], "God" :␣
↪x['Reference_Name']} for x in matchInfo]
    if (filtered[0]['Duration'] > 300 and filtered[0]['MMR'] > 0 and␣
↪filtered[len(filtered) - 1]['MMR'] > 0):
        writer.writerow([filtered[0]['Account_Level'], filtered[len(filtered) -␣
↪1]['Account_Level'], filtered[0]['MMR'], filtered[len(filtered) - 1]['MMR'],␣
↪findWinRate(filtered[0]['God']), findWinRate(filtered[len(filtered) -␣
↪1]['God']), filtered[0]['Duration']])

#
# This is more data cleanup
#

data = open('res.csv', 'r')
reader = csv.DictReader(data)

file = open('data.csv', 'w', newline='')
writer = csv.writer(file)
writer.writerow(["Level1", "Level2", "MMR1", "MMR2", "GodDiff", "Duration"])

for row in reader:
    god = abs(float(row['God1']) - float(row['God2']))
    writer.writerow([row['AccountLevel1'], row['AccountLevel2'],␣
↪row['MMR1'],row['MMR2'], god, float(row['Duration'])])

#
# This is all the code for the model
#

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

data = pd.read_csv('data.csv')

def poly_reg(X, y):
    # Split into training set
    X_train, X_rem, y_train, y_rem = train_test_split(
        X, y, test_size=0.5, random_state=42)

    # Split into validation and testing sets
    X_val, X_test, y_val, y_test = train_test_split(
        X_rem, y_rem, test_size=0.4, random_state=42)

    # Testing different degrees of poly and also basic linear regression
    degrees = [1, 2, 3, 5, 10]
    for degree in degrees:
        poly_reg = PolynomialFeatures(degree=degree, include_bias=False)
        lin_reg = LinearRegression()
        X_train_poly = poly_reg.fit_transform(X_train)
        lin_reg.fit(X_train_poly, y_train)

        print(f"Training error poly reg (deg {degree}):", mean_squared_error(
            y_train, lin_reg.predict(X_train_poly)))

        X_val_poly = poly_reg.fit_transform(X_val)
        print("Validation error:", mean_squared_error(
            y_val, lin_reg.predict(X_val_poly)))

        X_test_poly = poly_reg.fit_transform(X_test)
        print("Testing error:", mean_squared_error(
            y_test, lin_reg.predict(X_test_poly)))
        print("\n")

match_data = data.dropna()

# Features
X = np.array([
    match_data['Level1'],
    match_data['Level2'],
    match_data['MMR1'],
    match_data['MMR2'],
    match_data['GodDiff'],
]).T

# Label
y = np.array(match_data["Duration"])

poly_reg(X, y)
```