Project Outline

In this project, we will use the Trueskill approach to rank players in the game of Blood Bowl. We will build an online learning model to produce an overall ranking and a ranking with each of the races available in the game for each player. We build models based on different assumptions on the relations between races and draw rate and between race and results. Lastly we will expand the model to rank the races and explore possible insights about the performance of the races in a competitive context. To evaluate the performance we will use different models to predict the outcomes of games in a test set.
We use data from FUMBBL, which is an online community, where a digital version of the game have been played for about twenty years with data from an abundance of both historical and up to date games accessible.

Research Questions

- What is the skill level of a given player?
- What is the likely outcome of a game?
- To what extend does skill transfer between races?
- What is the power level of each race?
- What races performs best against the strongest races?
- How does different rules sets affect the dynamics of the game?

Data

As a starting point we have a large dataset with games from a variety of formats. While all the games in the dataset have been played with the newest version of the rules - published in 2020 - they are played with significantly different rules and formats. This data will however serve as a good starting point for building and testing the model we are building, and when we have a working model, we can scrape data specifically aimed at answering the specific research questions we are focusing on.
We have chosen a subset of the different columns in the subset to focus on. This is of course subject to change as we work on this project and learn more about the data and how it can affect our models. Initially we find the attributes regarding the match outcome, the race for each team and the ranking for the coach of each team the most important.

Descriptive Statistics

A notebook containing the descriptive statistics of our dataset has been added to the bottom of the file.

Model

To begin with, we start with a simple model based on the Trueskill model. See Figure 1 for a representation of our initial model. We will have some constants/hyperparameters that affect the players individual skills and how likely it is for the match to be a draw. There is of course some limitations in having these parts of the model as constants/hyperparameters instead of attributes to the model. Based on these constants, then the model will infer a skill and performance of each player which will model the result of the match.

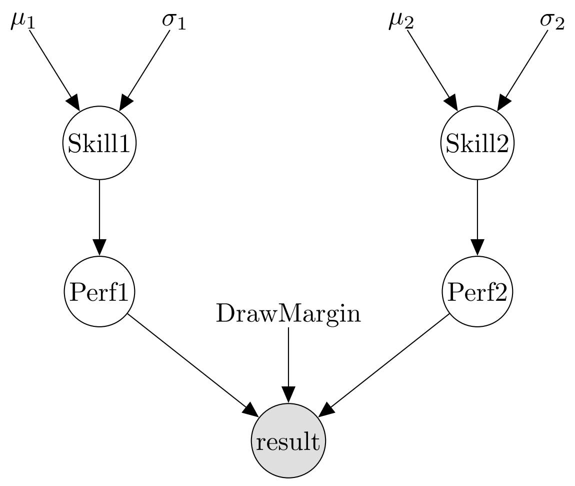


Figure 1: Initial model

Simple assumptions

Before starting working on implementing the model, we have discussed the following simple assumptions:

- Each player has constant skill levels.
- Skill levels follow a Gaussian distribution.
- The performance of a player is independent from game to game and follow a Gaussian distribution with with skill level as mean and a constant variance across all players.
- If the difference in performance in a given game is larger than a constant draw margin, the player with the highest performance wins, otherwise the game is a draw.

To also model changing skill levels over time and race specific skill levels we plan to introduce new assumptions and build a more suitable model.

Elaborate Assumptions

- Each player has skill levels with varying mean and constant variance.
- Each player has a skill level with each race and an overall skill level.
- The variance in skill level is constant for a given player, while the mean is variable.
- A players skill level in a given game depends on their skill levels in the previous game.
- The performance of a player is independent from game to game and symmetrically distributed in a bell shape.

In the first python cell, we store the data in a pandas dataframe. For initial cleanup, we will drop any rows with missing values. Then we select the columns that we expect will be used during this project.

In [ ]:
```python
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

path = "../df_matches.csv"

df = pd.read_csv(path, sep=',')

# Drop rows with missing values
df = df.dropna(subset=['match_date','match_conceded', 'coach1_CR', 'coach2_CR'])
relevant = ["match_id", "division_name", "match_date", "match_time", "match_conceded",
            "team1_coach_id", "team1_race_name", "team2_coach_id", "team2_race_name",
            "team1_score", "team2_score","team1_win","team2_win","mirror_match",
            "coach1_ranking","coach2_ranking","coach1_CR","coach2_CR"]
df = df[relevant]

df.head()
```

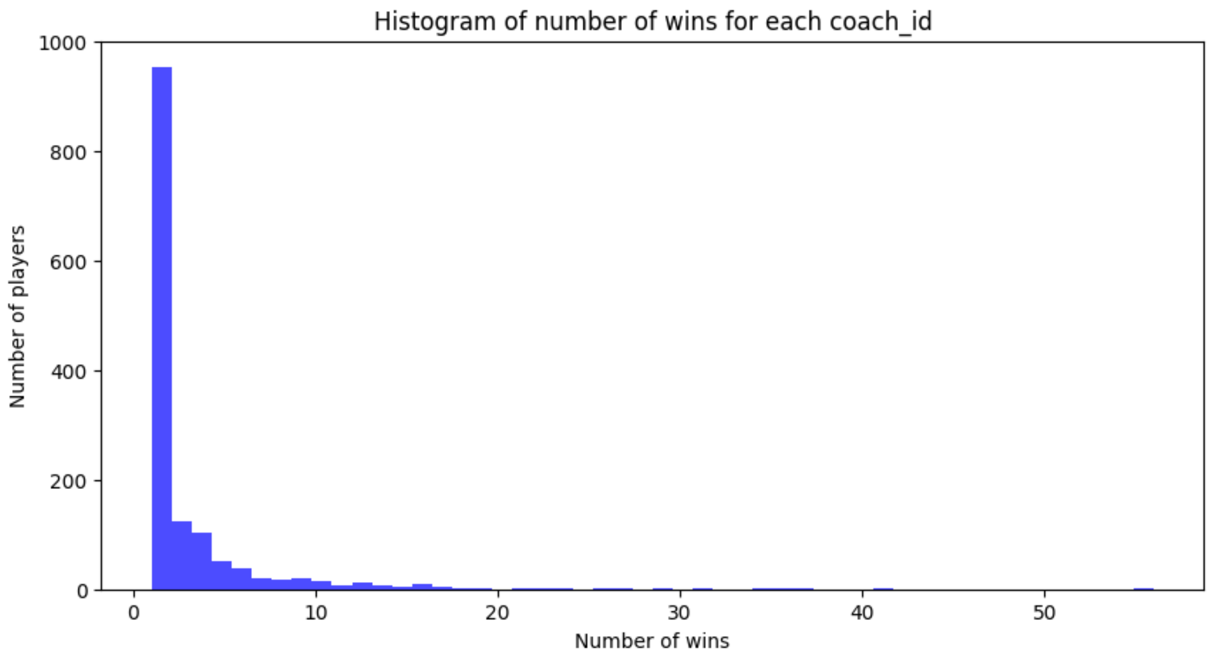
Out[ ]:

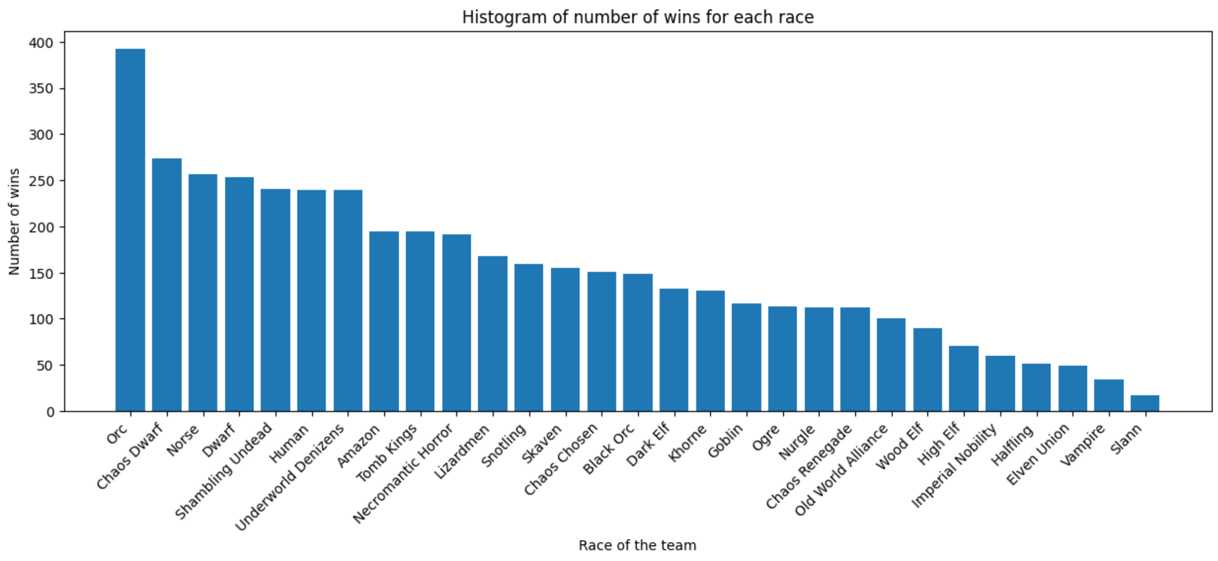
	match_id	division_name	match_date	match_time	match_conceded	team1_coach_id	team1_race_name	team
28	4216288	Blackbox	2020-08-01	23:54:36	Team2	240086	Chaos Dwarf	
86	4216346	Blackbox	2020-08-02	02:52:13	Team1	253902	Chaos Dwarf	
165	4216425	Blackbox	2020-08-02	11:22:53	Team1	245963	Dark Elf	
242	4216502	Blackbox	2020-08-02	16:23:36	Team1	253367	Chaos Chosen	
271	4216531	Ranked	2020-08-02	17:38:52	Team2	247586	Human	

Next we will investigate how some of the data is distributed. For the 8103 matches we have plotted how many wins each individual player has. We can see that the distribution is right-skewed, with most players having a small number of wins. We can also see that there are a few players with a large number of wins.
Next we plot the number of wins for each race in the dataset. We can see that some races definitely has more wins than others.

In [ ]:
```python
total_matches = df.shape[0]
print("Total matches: ", total_matches)

# make a new dataframe containing the unique coach ids and counts the number of times they won
coach1_wins = df[df["team1_win"] == 1].groupby("team1_coach_id").size().reset_index(name='wins')
coach2_wins = df[df["team2_win"] == 1].groupby("team2_coach_id").size().reset_index(name='wins')

# merge the dataframes and plot a histogram of number of wins for each coach_id
coach_wins = pd.concat([coach1_wins, coach2_wins])
coach_wins = coach_wins.groupby("team1_coach_id").sum().reset_index()
coach_wins = coach_wins.sort_values(by="wins", ascending=False)
coach_wins = coach_wins.reset_index(drop=True)

# plot a histogram of number of wins for each coach_id
plt.figure(figsize=(10,5))
plt.hist(coach_wins["wins"], bins=50, color='blue', alpha=0.7)
plt.xlabel("Number of wins")
plt.ylabel("Number of players")
plt.title("Histogram of number of wins for each coach_id")
plt.show()

# do the same but grouping by the race of the team instead of the coach
coach1_wins_race = df[df["team1_win"] == 1].groupby("team1_race_name").size().reset_index(name='wins')
coach2_wins_race = df[df["team2_win"] == 1].groupby("team2_race_name").size().reset_index(name='wins')

coach_wins_race = pd.concat([coach1_wins_race, coach2_wins_race])
coach_wins_race = coach_wins_race.groupby("team1_race_name").sum().reset_index()
coach_wins_race = coach_wins_race.sort_values(by="wins", ascending=False)
coach_wins_race = coach_wins_race.reset_index(drop=True)

plt.figure(figsize=(15,5))
plt.bar(coach_wins_race['team1_race_name'], coach_wins_race['wins'])
plt.xticks(rotation=45, ha='right')
plt.xlabel("Race of the team")
plt.ylabel("Number of wins")
plt.title("Histogram of number of wins for each race")
plt.show()
```

```
Total matches:  8103
```


Histogram of number of wins for each coach_id
Number of players
Number of wins


Histogram of number of wins for each race
Number of wins
Race of the team
Orc
Chaos Dwarf
Norse
Dwarf
Shambling Undead
Human
Underworld Denizens
Amazon
Tomb Kings
Necromantic Horror
Lizardmen
Snotling
Skaven
Chaos Chosen
Black Orc
Dark Elf
Khorne
Goblin
Ogre
Nurgle
Chaos Renegade
Old World Alliance
Wood Elf
High Elf
Imperial Nobility
Halfling
Elven Union
Vampire
Slann