Advanced probabilistic machine learning
Mini-project

Stefanos Tsampanakis
Department of Information Technology
Uppsala University
`stefanos.tsabanakis@gmail.com`

Christos Pilichos
Department of Information Technology
Uppsala University
`christos.pilichos.9957@student.uu.se`

Vasileios Toumpanakis
Department of Information Technology
Uppsala University
`billtoubanakis@yahoo.gr`

Marius Vangeli
Department of Information Technology
Uppsala University
`Marius.vangeli.8733@student.uu.se`

1 Introduction

In this project, we are interested in evaluating the skills of participants in a competition, by analyzing the results of the matches between pairs of players or teams. We focus on a probabilistic machine learning approach. To this end we apply a probabilistic model based on the TrueSkill ranking system , such that the skill level of every player, as well as the outcome of the game are represented as Gaussian random variables, while the result of the game is expressed as a discrete random variable. For the development of our Trueskill method, the *SerieA.csv* dataset containing the results of the Italian Serie A of 2018-2019 is utilized and then is tested for the online competitive game 'Counter Strike-Global offensive' dataset.

2 Modeling

2.1

Our Bayesian model consists of three gaussian random variables s_1, s_2 and t, as well as one discrete variable y. The hyperparameters in our model consists of the mean and variance of s_1 and s_2 (μ_{s1}, μ_{s2}, σ^2_{s1} and σ^2_{s2}), and the variance of t (σ^2_t). The mean of t is defined as:

$$\mu_t = s_1 - s_2$$

There is thus a total of 5 hyperparameters in our model. The variables s_1 and s_2 signify the skill levels of two competing players, and the variable t indicates the performance difference between the two players in a given game, determined by the skills s_1 and s_2 of the two players. The discrete random variable y, signify the outcome of a given game and can only take the values 1 or -1, with the preliminary assumption that there is no possibility of draw. We present the probability distributions associated with these variables as follows.

$$p(s_1) = \mathcal{N}(\mu_1, \sigma_1^2) \text{ and } p(s_2) = \mathcal{N}(\mu_2, \sigma_2^2)$$

$$p(t|s_1, s_2) = \mathcal{N}(s_1 - s_2, \sigma_3^2) \text{ and } y = sign(t) = \begin{cases} 1 & \text{if } t > 0 \\ -1 & \text{if } t < 0 \end{cases}$$

Finally our model can be represented as a joint distribution:

$p(s1, s2, t, y) = p(y|t)p(t|s1, s2)p(s1)p(s2)$

2.2

Using the Bayesian graph to factor the joint distribution we have that:

$$p(s_1, y|t) = \frac{p(s_1, y, t)}{p(t)} = \frac{p(y|t)p(t|s_1)p(s_1)}{p(t)} = p(y|t)\frac{p(t|s_1)p(s_1)}{p(t)} = p(y|t)p(s_1|t) \tag{1}$$

We thus have $p(s_1, y|t) = p(y|t)p(s_1|t) \Rightarrow s_1 \perp y \mid t$.

Similarly for $s_2 \perp y \mid t$.

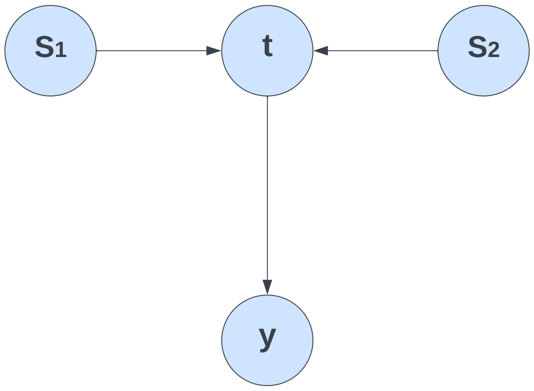


Figure 1: Factor graph

3 Bayesian Network

3.1 Computing with the model

Using Bayes' Theorem, we demonstrate the conditional probabilities of variables s_1 and s_2 when variables t and y are known, as follows

$$p(s_1, s_2|t, y) = \frac{p(s_1,s_2,t,y)}{p(t,y)} = \frac{p(y|t)p(t|s_1,s_2)p(s_1)p(s_2)}{p(y|t)p(t)} = p(s_1, s_2|t)$$

We will use Corollary 1, so first we formulate the distributions $p(t|s_1, s_2)$ and $p(s_1, s_2)$ appropriately.

In terms of Cor.1 $x_a = \begin{pmatrix} s_1 \\ s_2 \end{pmatrix}$ and $x_b = t$. So,

$$p(t|s_1, s_2) = \mathcal{N}(t; s_1 - s_2, \sigma_3^2) = \mathcal{N}(t; (1 \quad -1)\begin{pmatrix} s_1 \\ s_2 \end{pmatrix}, \sigma_3^2) \tag{2}$$

, that is $A = (1 \quad -1)$ and $\Sigma_{t|s_1,s_2} = \sigma_3^2$.

For (s_1, s_2), they are independent (without t being observed), so their joint distribution would be a 2D Gaussian where the covariance matrix is diagonal (because of their independence) with their respective variances as values. We have,

$$p(s_1, s_2) = \mathcal{N}(\begin{pmatrix} s_1 \\ s_2 \end{pmatrix}; \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \begin{pmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{pmatrix}) \tag{3}$$

,that is $\Sigma_{s_1,s_2} = \begin{pmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{pmatrix}$ and $\mu_{s_1,s_2} = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}$.

Based on the above formulation and Corollary 1, we have,

$$p(s_1, s_2|t) = \mathcal{N}(\begin{pmatrix} s_1 \\ s_2 \end{pmatrix}; \mu_{s_1,s_2|t}, \Sigma_{s_1,s_2|t}) \tag{4}$$

, where:

$$\Sigma_{s_1,s_2|t} = (\Sigma_{s_1,s_2}^{-1} + A^T \Sigma_{t|s_1,s_2}^{-1} A)^{-1} = \frac{1}{\frac{(\sigma_1^2+\sigma_3^2)(\sigma_2^2+\sigma_3^2)}{(\sigma_1\sigma_2\sigma_3)^2} - \frac{1}{\sigma_3^2}} \begin{pmatrix} 1 + (\frac{\sigma_3^2}{\sigma_2^2})^2 & 1 \\ 1 & 1 + (\frac{\sigma_3^2}{\sigma_1^2})^2 \end{pmatrix} \tag{5}$$

$$\mu_{s_1,s_2|t} = \Sigma_{s_1,s_2|t}(\Sigma_{s_1,s_2}^{-1}\mu_{s1,s2} + A^T \Sigma_{t|s_1,s_2}^{-1} t) = \frac{1}{\frac{(\sigma_1^2+\sigma_3^2)(\sigma_2^2+\sigma_3^2)}{(\sigma_1\sigma_2\sigma_3)^2} - \frac{1}{\sigma_3^2}} \begin{pmatrix} (1 + (\frac{\sigma_3}{\sigma_2})^2)(\frac{\mu_1}{\sigma_1^2} + \frac{t}{\sigma_3^2}) + (\frac{\mu_2}{\sigma_2^2} - \frac{t}{\sigma_3^2}) \\ (\frac{\mu_1}{\sigma_2^2} + \frac{t}{\sigma_3^2}) + (1 + (\frac{\sigma_3}{\sigma_2})^2)(\frac{\mu_2}{\sigma_2^2} - \frac{t}{\sigma_3^2}) \end{pmatrix} \tag{6}$$

3.2 Calculating $p(t|y, s1, s2)$

$$p(t|s_1, s_2, y) = \frac{p(t, s_1, s_2, y)}{p(s_1, s_2, y)} = \frac{p(y|t)p(t|s_1, s_2)p(s_1, s_2)}{p(s_1, s_2, y)} = C \cdot p(y|t)p(t|s_1, s_2) \propto p(y|t)p(t|s_1, s_2)$$

For every t, y:

$$p(y|t) = \begin{cases} 1, & y = 1, t \geq 0 \\ 0, & y = -1, t \geq 0 \\ 0, & y = 1, t < 0 \\ 1, & y = -1, t < 0 \end{cases}$$

Thus for every t, s_1, s_2, y, we have:

$$p(t|s_1, s_2, y) = C \cdot \mathcal{N}(t; s_1 - s_2, \sigma_3^2) \cdot \begin{cases} 1, & y = 1, t \geq 0 \\ 0, & y = -1, t \geq 0 \\ 0, & y = 1, t < 0 \\ 1, & y = -1, t < 0 \end{cases} = \begin{cases} TruncN(t; s_1 - s_2, \sigma_3^2, [0, +\infty)), & y = 1 \\ \\ TruncN(t; s_1 - s_2, \sigma_3^2, (-\infty, 0), & y = -1 \end{cases}$$

3.3 Calculating $p(y = 1)$

The marginal probability $p(y = 1)$ can be calculated as $p(t > 0)$.

Using Corollary 2 on $p(t|s_1, s_2) = \mathcal{N}(t; s_1 - s_2, \sigma_3^2)$ and $p(s_1, s_2) = \mathcal{N}(\begin{pmatrix} s_1 \\ s_2 \end{pmatrix}; \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \begin{pmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{pmatrix})$, we obtain that the marginal distribution of t is Gaussian. Specifically, $p(t) = \mathcal{N}(t; \mu_1 - \mu_2, \sigma_1^2 + \sigma_2^2 + \sigma_3^2)$.

The Z-score for $t = 0$ is given by $Z = \frac{(t-\mu_t)}{\sqrt{\sigma_t^2}}$.

Finally, $p(t > 0)$ can be found using the standard normal distribution table by $1 - \Phi(Z)$.

4 Gibbs sampler

We implement a Gibbs sampler to calculate the posterior distribution for the skills s_1 and s_2, given the result of the match y. We start by assuming the same prior distribution for the two players by choosing $\mu_{s_1} = 20, \mu_{s_2} = 30$ and $\sigma_{s_1}^2 = \sigma_{s_2}^2 = 1$ for the mean and variance values respectively. Then for a fixed number of samples and in case that Player 1 only wins ($y = 1$) the subsequent iterated process unfolds as follows:

- Computation of $p(t > 0|s1, s2)$
- Computation of $p(s1, s2|t > 0)$
- Draw samples s_1 and s_2 from multivariate normal

For a number of 4000 samples now the posterior distribution $p(s1, s2|y)$ is plotted.

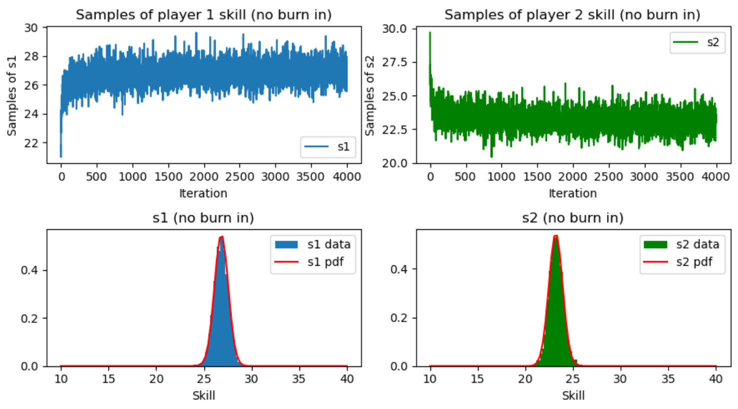


Figure 2: Posterior distribution $p(s1, s2|y = 1)$ without burn in

Given the outcome of the game (y=1), we expect the skill of player 1 to go up and player 2 to go down, which we can see in resulting distributions in Fig.(2). By running several experiments with different initial hyperparameters we were able to see a general trend of burn-in ≈ 250 iterations. Then, in order to evaluate our burn-in choice we re-run the same experiment and we get the following results:

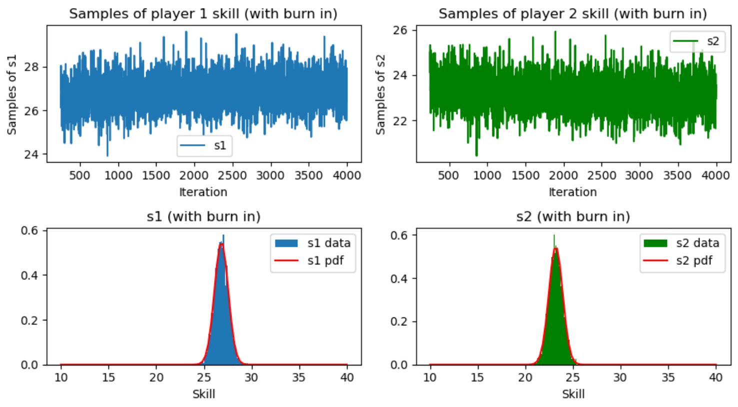
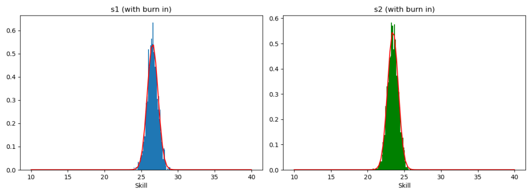


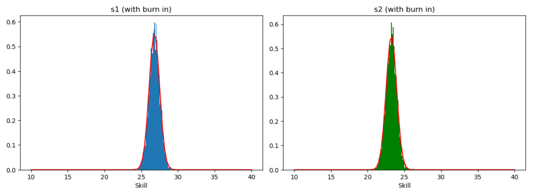
Figure 3: Posterior distribution $p(s1, s2|y = 1)$ after burn in

We can now see that the samples, which had a strong dependence on the initial conditions and were distant from the stationary distribution, have been effectively eliminated. Therefore, we can confidently conclude that our choice of burn-in period was successful.

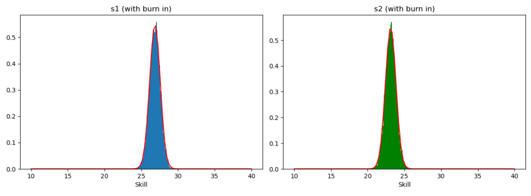
The next part of our research involves determining an optimal sample size that strikes a balance between estimation accuracy and computational efficiency. In pursuit of this objective, we calculate the execution times for four different sample sizes and generate the corresponding histograms.



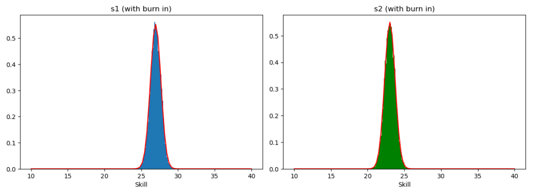
(a) 2000 samples/ t=0.9sec



(b) 4000 samples/ t=1.7sec



(c) 10000 samples/ t=4.3sec



(d) 14000 samples/ t=5.5sec

Figure 4: Histogram for a varying number of samples and the execution time

Given that the serieA dataset consists of 380 games, we can estimate the run time of running Gibbs over the entire dataset to be 6, 11, 27 and 35 minutes respectively for samples sizes in Fig.(4). We have opted for 2,000 samples for our Gibbs sampler as it has a relatively short runtime while still resulting in a good approximation. This decision to prioritize runtime will be even more decisive later as we implement our new dataset with over 3000 games. This choice ensures both accuracy, as confirmed by the well-fitted Gaussian, and a reasonable execution time.
Finally, we compare the prior distribution with the Gaussian approximation of the posterior distribution for both Player's skills, s1 and s2, resulting in the following outcomes.

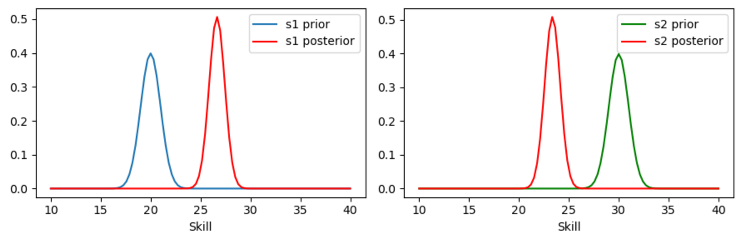


Figure 5: Prior-Posterior distribution

Again, as expected, we see that the skill of player 1 shifts to the right while the skill of player 2 shifts to the left, due to the fact that player 1 wins the game.

5 Assumed Density Filtering

In this part, we use Gibbs sampling to iterate over all games. The posterior distributions derived from the previous match between 2 teams serves as the prior for the next, i.e *assumed density filtering (ADF)*. It's important to note that we exclude any draw outcomes or even the goal-difference from the data set before performing these computations. We proceed to examine the matches within the SerieA dataset and make skill estimations for all the teams encompassed in the dataset. This yields a conclusive ranking of the teams 1.

To investigate the impact of match order on the ultimate team skill rankings, we shuffle the dataset and conduct a new run of the ADF 2.

If we compare Table 1 and Table 2 it becomes evident that ADF is really sensitive to the order of the matches. It seems that winning or losing the first games has a more pronounced influence on a team's skill level compared to winning a later match. Additionally, we know that the current distribution at any given moment always influences the posterior distribution, so shuffling the gaming dates results in different distributions.

6 Using the model for predictions

A new function called *prediction* is now introduced in our ADF model to perform one-step-ahead predictions of the games. Draws are excluded from the dataset in this part so the only possible outcomes are: player 1 wins (1) or player 2 wins (-1). This function simply predicts y=1 or y=-1 based on which of the two teams has the largest mean skill.
The prediction rate of this method found to be approximately 64%, which is higher than random guessing (50%).

7 Factor graph

We have

$$fs_1 = \mathcal{N}(\mu_1, \sigma_1^2)$$
$$fs_2 = \mathcal{N}(\mu_2, \sigma_2^2)$$
$$f_{s_1 s_2 t} = \mathcal{N}(s_1 - s_2, \sigma_3^2)$$
$$fty = sign(t)$$

For the messages we have:

$$\mu_1 = \mu_2 = fs_1 = \mathcal{N}(\mu_1, \sigma_1^2)$$
$$\mu_4 = \mu_3 = fs_3 = \mathcal{N}(\mu_2, \sigma_2^2)$$
$$\mu_6 = \int f_{s_1 s_2 t}(t)\mu_2\mu_4 \, ds_1 ds_2 = \mathcal{N}(t, \mu_1 - \mu_2, \sigma_1^2 + \sigma_2^2 + \sigma_3^2)\text{(Collorary 2)}$$
$$\mu_7 = \delta(y = y_{obs}), \mu_8 = \delta(sign(t) = y)\delta(y = y_{obs}) = \delta(sign(t) = y_{obs})$$
$$p(t|y = y_{obs}) \propto \mu_8\mu_6 = \delta(sign(t) = y_{obs})\mathcal{N}(t, \mu_1 - \mu_2, \sigma_1^2 + \sigma_2^2 + \sigma_3^2)$$

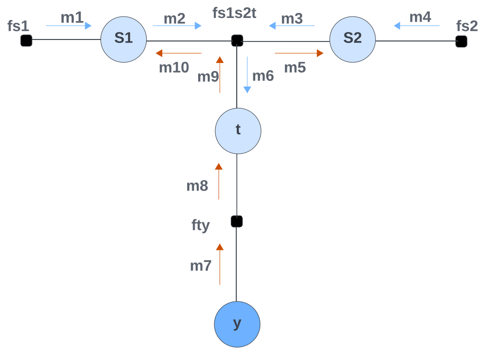


Figure 6: Factor graph

8 A message-passing algorithm

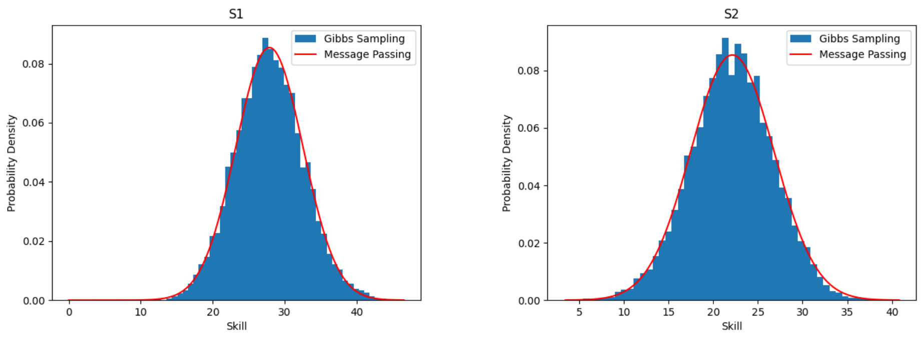


Figure 7: Posterior distirbutions

The messages computed in section 8 are then implemented in an message passing algorithm to compute the posterior distributions of s_1 and s_2. To accomplish that we used moment matching.

We approximate p(tly) with $q(t) = \mathcal{N}(t, m_t, \sigma_t^2)$ so for the outgoing messages we have

$$\mu_9 = \frac{q(t)}{\mu_6} = \frac{\mathcal{N}(t,m_t,\sigma_t^2)}{\mathcal{N}(t,\mu_1-\mu_2,\sigma_1^2+\sigma_2^2+\sigma_3^2)} = \mathcal{N}(t, \frac{m_t(\sigma_1^2+\sigma_2^2+\sigma_3^2)-(\mu_1-\mu 2)\sigma_t^2}{\sigma_1^2+\sigma_2^2+\sigma_3^2-\sigma_t^2}, \frac{\sigma_t^2(\sigma_1^2+\sigma_2^2+\sigma_3^2)}{(\sigma_1^2+\sigma_2^2+\sigma_3^2)-\sigma_t^2}) = \mathcal{N}(t, m_9, \sigma_9^2)$$

$$\mu_5 = \int f_{s_1 s_2 t}(t)\mu_9\mu_2 \, dt = \int \mathcal{N}(t, m_t, s_t)\mu_2 dt = \mathcal{N}(t, \mu_2 - m_9, \sigma_1^2 + \sigma_3^2 + \sigma_9^2)$$

$$\mu_{10} = \int f_{s_1 s_2 t}(t)\mu_9\mu_3 \, dt = \int \mathcal{N}(t, m_t, s_t)\mu_1 dt = \mathcal{N}(t, \mu_3 + m_9, \sigma_2^2 + \sigma_3^2 + \sigma_9^2)$$

$$p(s1|t) = \mu_{10}\mu_1$$

$$p(s2|t) = \mu_5\mu_4$$

In Figure(7) we can see two different ways of looking at the skill distribution after a single match, where Player 1 wins. One approach uses a message passing algorithm, and the other relies on a Gaussian approximation from Gibbs sampling. We can see that both methods give similar posterior results for the same match.

9 Our own data

Assumed Density Filtering is now tested for a new data set. Specifically, we present the skill ranking of 47 Counter Strike (CSGO) teams based on our model for 3401 total games played between 2018-11-11 and 2020-03-18 3. This dataset was downloaded from kaggle and originally included more than 45000 games played by 654 teams during the period 2015-2020. The selection of the 47 teams was made by choosing a shorter time frame and removing all teams with less than 200 total games played and then only including the games played between these teams. The prediction rate for the CSGO games was only 60%.

10 Project extension

Finally, we introduce the three following extensions:

1. We incorporate the score difference in the game, granting a larger skill boost to the team who wins with a large score lead. This is achieved by increasing the skill difference by adding a parameter which is proportional to the score difference: $\mu_t = (s_1 - s_2) + score\_parameter$. This increased the accuracy of the predictions to 66%.
2. We account for a slight advantage held by the home team by giving a larger skill boost to a team winning an away game and a smaller skill boost for a team winning a home game. This

extension did not significantly change the accuracy of the predictions for better or worse. Something that did improve our accuracy however was changing the prediction function to favour the home team if both teams were evenly matched. This improved our accuracy to 67%.

3. We incorporate the possibility of draws in the Gibbs sampler by adding the special case of y = 0. For a draw we want the skill difference to shrink. To achieve this we sample t from the full normal distribution instead of the truncated one, and with a smaller mean of t: $\mu_t = (s_1 - s_2) \cdot 0.9$. This ensures that the new skills of the teams will be proportional to their original skills, but slightly closer to each other.

The new ranking of the football teams after the implementation of all the extensions can be seen in the Appendix (4) The One-Step-Ahead algorithm still can only predict win or loose, meaning that all draws will be predicted wrongly. Still, the accuracy is 48%, which is better than random guessing: 33% (outcome can be 1, -1 or 0). If we instead apply the above mentioned extensions but don't include the draws in our predictions, we get an accuracy of 68%

11 Appendix

Table 1: Teams after the season, no shuffle

Team	Mean	Variance
Juventus	29.964	0.231
Roma	29.300	0.081
Atalanta	28.708	0.034
Milan	27.904	0.058
Inter	27.589	0.025
Torino	27.302	0.071
Bologna	26.997	0.171
Napoli	26.885	1.119
Sampdoria	26.334	2.085
Lazio	25.037	0.067
Udinese	24.992	0.037
Empoli	24.689	0.372
Sassuolo	23.898	0.024
Spal	23.233	0.143
Parma	22.613	0.050
Cagliari	22.526	0.104
Genoa	22.340	0.037
Chievo	21.633	0.017
Frosinone	21.335	0.027
Fiorentina	20.562	0.060

Table 2: Teams after the season, with shuffle

Team	Mean	Variance
Torino	29.786	0.0075
Napoli	29.776	0.028
Juventus	29.622	0.011
Bologna	29.111	0.183
Fiorentina	28.873	0.035
Cagliari	28.537	0.055
Roma	28.413	0.041
Atalanta	28.265	0.040
Milan	27.447	0.017
Inter	27.418	0.301
Sassuolo	26.401	0.054
Lazio	26.230	0.130
Genoa	26.100	0.017
Empoli	24.817	0.071
Spal	24.722	0.039
Sampdoria	24.280	0.091
Parma	23.741	0.048
Udinese	23.322	0.018
Chievo	22.710	0.012
Frosinone	21.981	0.127

Table 3: Mean and Variance for Teams

Team	Mean	Variance
HAVU	45.399050	0.252836
fnatic	43.983383	0.349155
Valiance	42.126143	0.146061
Astralis	41.460824	0.095420
MIBR	37.969604	0.230447
North	37.884376	0.838966
Spirit	37.821397	0.095173
NiP	37.020101	0.714597
SJ	34.948453	0.221274
Windigo	34.555075	0.048858
Sprout	33.304795	0.033404
Winstrike	33.097466	0.108520
mousesports	33.020734	0.266044
pro100	32.673950	0.347085
Chiefs	32.653549	0.255527
TYLOO	32.186936	0.072056
Liquid	31.966211	0.095065
AVANGAR	31.737960	0.049332
Copenhagen Flames	30.461015	0.207920
FURIA	28.922990	0.181809
HellRaisers	28.869383	0.106598
G2	28.496939	0.213666
ENCE	28.029786	0.092452
BIG	27.139374	0.138528
INTZ	26.490026	0.165983
Heroic	26.442339	0.454055
forZe	25.989427	0.100402
FaZe	25.981548	0.326603
Grayhound	25.847888	0.044797
Heretics	25.684713	0.293121
TeamOne	24.133485	0.126474
Vitality	23.969702	0.152724
Tricked	23.936855	0.042072
Chaos	23.727558	0.427929
Unique	23.361581	0.083469
DETONA	22.923449	0.179727
AGO	22.061743	0.320605
Movistar Riders	21.059525	0.365861
Nemiga	20.552426	0.156954
GamerLegion	20.309886	0.190049
Singularity	19.367728	0.043788
ORDER	18.753775	0.106598
PACT	18.665779	0.098206
ALTERNATE aTTaX	18.636612	0.023402
W7M	17.267472	0.127036
DreamEaters	15.797202	0.075990
Virtus.pro	7.636134	0.033774

Table 4: Mean and Variance for Football Teams

Team	Mean	Variance
Atalanta	36.253020	0.460295
Bologna	30.940821	0.365403
Roma	29.714629	0.281760
Napoli	28.578327	0.466831
Inter	28.464694	0.630568
Milan	28.065953	0.134532
Torino	28.003617	1.035826
Udinese	26.442018	0.195860
Empoli	24.807333	0.678466
Spal	23.349055	0.157157
Lazio	22.053444	0.612249
Cagliari	21.300309	0.165333
Juventus	20.234150	1.612227
Sampdoria	19.945430	1.204770
Sassuolo	15.449148	0.190218
Parma	15.089464	0.271466
Genoa	13.899693	0.092526
Fiorentina	13.295344	0.096667
Chievo	12.438943	0.131034
Frosinone	10.595734	0.170006