Математическая модель для определения лучшего спортсмена в разных видах спорта.

•••

Проект выполнили: Лазарев Андрей, Авраменко Денис, Баданов Чингис, Рудась Дарья

Математическая часть

Математическая часть, хранение данных

Выборка - некоторая таблица, имеющая следующие поля - вектор с рангами игроков, матрица с набранными очками, а также вектор смещений рангов. Весь алгоритм строится вокруг этой таблицы.

Далее представлена таблица, которая является примером выборки.

Математическая часть, пример выборки

Name	Rank	P1	P2	Final
Bill	50.0	5	3	49.9857 ~ 50
Bob	50.0	6	1	49.4853 ~ 49
Andrew	50.0	1	3	43.6951 ~ 44
Jack	50.0	5	7	55.1611 ~ 55
Ronny	50.0	8	3	52.8022 ~ 53

Математическая часть, основа функции смещений рангов

За основу мы взяли функцию следующего вида вида, ***x*** *- количество очков, которые набрал игрок,* а само значение функции - *смещение ранга этого игрока.*

Параметр ***a*** *- центральное значение очков,* а параметр ***k*** *- наклон функции,* этот параметр должен зависеть от стандартного отклонения рангов игроков, а также от отношения разницы максимального и центрального кол-ва очков и разницы центрального и минимального.

$$f(x) = k(x - a)^p$$

Математическая часть, вычисление параметра *p*

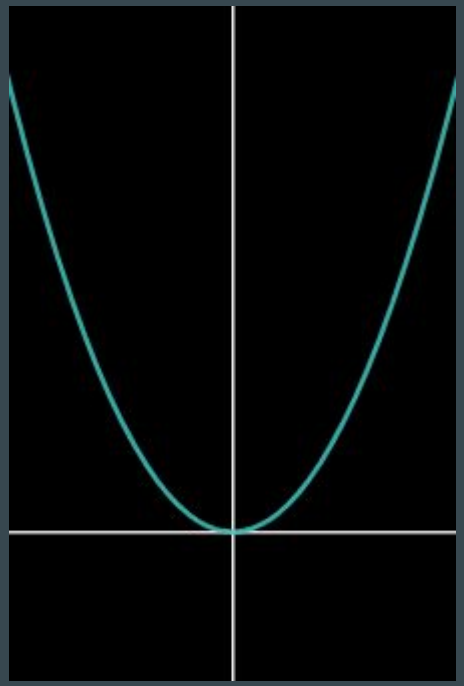
Параметр ***p*** мы определили путем написания алгоритма, который генерировал случайных участников со случайными рангами и набранными очками, параметр k этом алгоритме мы приняли за единицу. Мы подставили туда три разных значений степени - 1 (линейная), 2 (квадратичная), 3 (кубическая).

Смещения рангов, вычисленных с помощью квадратичной зависимости, нам показались более подходящими для конкретных значений набранных очков. Поэтому мы остановились на квадратичной зависимости

Математическая часть, отражение отдельной части функции

Нам нужно чтобы та часть функции, принадлежащая промежутку от бесконечности до *a*, была отображена по оси *y,* поэтому мы добавляем выражение следующего вида:

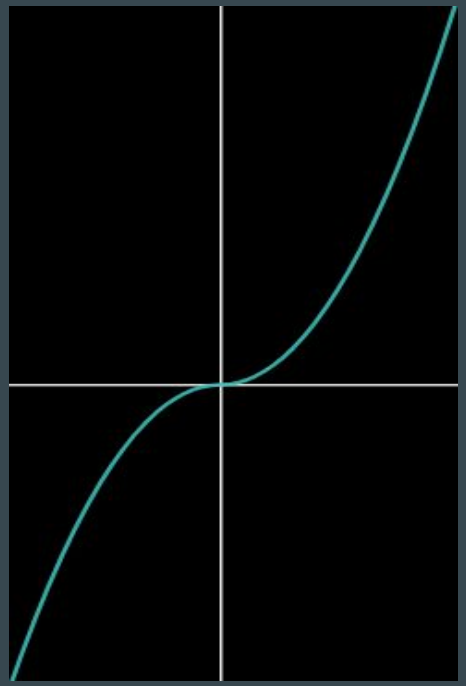
$$g(x) = \frac{x-a}{|x-a|}$$



Математическая часть, финальный вид функции смещения

$$bias_i = f(points_i) * g(points_i)$$

$$bias_i = k(points_i - ponts_{central})^2 \left(\frac{points_i - ponts_{central}}{|points_i - ponts_{central}|} \right)$$



Математическая часть, “приоритет”

Очки в каждом спорте могут считаться по-разному, но для нашей модели подходит только формат, где очки игроков входят в промежуток от нуля (вкл.) до бесконечности, а также модель обрабатывает 0, как наихудший результат.

Поэтому, чтобы универсализировать нашу модель для разных систем подсчета очков, мы добавили так называемый приоритет.

Математическая часть, “приоритет”

Приоритет - одна или несколько функций, которые присутствуют в выборке наравне с матрицей полученных очков. После суммирования всех очков из матрицы в вектор *(про алгоритм - далее)* к каждому элементу вектора применяется приоритет. Если в модели задана всего одна функция приоритета, то ко всем элементам применяется она. Если же задан массив (он должен быть такой же длины, что и количество игроков), то для каждого значения вектора применяется соответствующая функция приоритета из массива.

Алгоритмическая часть

Алгоритмическая часть, используемые технологии

Алгоритм был написан на языке Java, без сторонних библиотек. Мы использовали среду разработки от JetBrains.

Помимо этого, мы написали собственную библиотеку для векторных и матричных операций (скалярное произведение, сложение векторов и т.д.), чтобы в будущем было проще масштабировать программу.

Алгоритмическая часть, алгоритм

Ниже кратко представлен алгоритм нахождения смещений рангов в конкретной выборке.

1) Преобразовать матрицу очков в вектор, суммировав столбцы.
2) Применить к вектору с очками функцию/функции приоритета.
3) Найти показатели, которые нужны для вычисления угла наклона функции *(ранг_макс, ранг_мин, ранг_центр, стандартное отклонение по очкам)*.
4) Вычислить K и подставить значение в функцию-активатор (мы использовали модифицированный арктангенс), чтобы в разных выборках угол наклона всегда был в одном промежутке.
5) Подставить K в функцию и посчитать смещения рангов.

Плюсы и минусы алгоритма

Плюсы

- Кросс-платформенность *(так как мы писали на Java)*
- Масштабируемость *(мы использовали самописные библиотеки, а значит можно в любой момент добавить к ним функций)*
- Универсальность за счет “приоритета”

Минусы

- Не самая высокая точность модели *(следует из универсальности)*
- Отсутствие какого-либо графического представления *(например, интерфейса)*

Практический смысл модели

Практический смысл модели

Программа сможет помочь в будущем спортивным порталам, а также простым людям.

Вряд ли большинство людей, которые не относят себя к фанатам, захотят смотреть все матчи какого-то чемпионата, для них и создана эта модель: они просто посмотрят статистику лучших игроков и посмотрят матчи с ними, не зная результат самих матчей (модель просто “говорит” рейтинг игрока, определить что-либо по которому сложно).