Predicting the Maths Classes That People Will Attend in the IB

Adrian Fang

June 30, 2023

1 Introduction

Predicting maths classes for students in the next year is not as intuitive as it seems. Given how classes do not entirely follow a direct route, some logic and an algorithm has to be implemented to be able to create accurate predictions to recommend classes for students in the next year.

2 Leveraging Data to Establish Linear Relationships

2.1 Linear Functions

Maths classes do a follow a linear trend in predicting classes of a certain difficulty. For instance, a class that is not very difficult in the MYP would yield an easy class for the IB, with a similar case for classes with higher difficulty. Suppose that all data has already been converted into numeric values, which means that a line can be obtained using existing data to predict further values. To approximate this line, every single combination of point pairs can calculate a specific line, where the distance between real values to their predicted value from the line can be averaged, such that the lowest average will imply the most accurate line for that set of points. Suppose that points A and B represent the point pairs, below:

$$A\left(x_1, y_1\right) \tag{1}$$

$$B\left(x_2, y_2\right) \tag{2}$$

Since the relationship is assumed to be linear, the gradient can be obtained using the slope formula:

$$m = \frac{y_2 - y_1}{x_2 - x_1} \tag{3}$$

To obtain the constant of the linear function, consider its equation in slope intercept form:

$$y = mx + b \tag{4}$$

Since two points of the line are theoretically known, one of the points can be inserted into the equation. To solve for the constant, b, mx has to be subtracted from both sides of the equation:

$$b = y - mx \tag{5}$$

Now the linear function has been solved, and can be used by the algorithm to make predictions.

2.2 Averaging Distances and Finding the Best Line

Given that there are an infinite amount of distances between an external point to a linear function, a relationship needs to be established to obtain the minimum distance. To find the distance between a point and the solved line, the perpendicular line passing through the point has to be found. Since the Pythagorean Theorem states that $a^2 + b^2 = c^2$, a 90 degree angle created by the lines imply that it would theoretically be a leg of a right triangle rather than the hypotenuse: which suggests that its distance has been minimised. The intersection point between the two different lines is then used to find the distance between said point and the target point. Consider the following proof, which utilises the distance formula along with optimisation. The distance of a point to a line can expressed below:

$$D^2 = (a - x)^2 + (b - y)^2 \tag{6}$$

Variables a and b represent the external point and x representing the x coordinate of a point on the line, where y is a function of x. Since y is a linear function, it can be reasonably inferred that $\frac{dy}{dx} = m$, where m is the gradient of the line. Differentiating D the done:

$$2D\frac{dD}{dx} = \frac{d}{dx}\left((a - x)^2 + (b - y)^2\right) \tag{7}$$

$$2D\frac{dD}{dx} = 2(a - x) \cdot -1 + 2(b - y) \cdot -\frac{dy}{dx} \tag{8}$$

$$D\frac{dD}{dx} = -(a - x) - \frac{dy}{dx}(b - y) \tag{9}$$

Given that $\frac{dy}{dx} = m$ and $\frac{dD}{dx} = 0$, the equation is simplified further:

$$0 = -(a - x) - m(b - y) \tag{10}$$

$$m(b - y) = x - a \tag{11}$$

As y is a linear function where $\frac{dy}{dx} = m$, its original function is $mx + c$, where c is a constant.

$$m\left(b - (mx + c)\right) = x - a \tag{12}$$

$$mb - m^2x - mc = x - a \tag{13}$$

$$mb - mc + a = x + m^2x \tag{14}$$

$$mb - mc + a = x\left(1 + m^2\right) \tag{15}$$

$$x = \frac{mb - mc + a}{1 + m^2} \tag{16}$$

With an equation for x, the linear function can be used to solve for y:

$$y = m\left(\frac{mb - mc + a}{1 + m^2}\right) + c \tag{17}$$

$$y = \frac{m^2b - m^2c + am}{1 + m^2} + c \tag{18}$$

Since x and y are known in terms of known variables, to prove that the slope is equivalent to the perpendicular slope of m (which is $-\frac{1}{m}$), the slope formula can be used. Let m_2 be the slope of the line created by $P(a, b)$ and $Q(x, y)$.

$$m_2 = \frac{b - \left(\frac{m^2b - m^2c + am}{1+m^2} + c\right)}{a - \frac{mb - mc + a}{1+m^2}} \tag{19}$$

$$m_2 = \frac{\frac{b\left(1+m^2\right) - \left(ma + m^2b - cm^2\right) - c\left(1+m^2\right)}{1+m^2}}{\frac{a(1+m^2) - (a+mb-cm)}{1+m^2}} \tag{20}$$

$$m_2 = \frac{b\left(1 + m^2\right) - \left(ma + m^2b - cm^2\right) - c\left(1 + m^2\right)}{a\left(1 + m^2\right) - (a + mb - cm)} \tag{21}$$

$$m_2 = \frac{b + m^2b - ma - m^2b + cm^2 - c + cm^2}{a + am^2 - a - mb + cm} \tag{22}$$

$$m_2 = \frac{b - ma - c}{am^2 - mb + cm} \tag{23}$$

$$m_2 = \frac{b - ma - c}{-m\left(b - am - c\right)} \tag{24}$$

$$m_2 = -\frac{1}{m} \tag{25}$$

Now that the slope has been proven to be perpendicular, a simpler equation for x can be found to be used in the algorithm. Suppose that the perpendicular line is defined as $y_2 = m_2x + k$, where k is a constant. k can be obtained using the operations in Sec. 2.1. These two lines are shown below:

$$y = mx + c \tag{26}$$

$$y_2 = m_2x + k \tag{27}$$

In the event of an intersection, their coordinates would be the same. This can be expressed by equating the two equations to one another.

$$mx + c = m_2x + k \tag{28}$$

$$mx - m_2 x = k - c \tag{29}$$

$$x\left(m - m_2\right) = k - c \tag{30}$$

$$x = \frac{k - c}{m - m_2} \tag{31}$$

The method to find the y coordinate is the same as previously.

$$y = m\left(\frac{k - c}{m - m_2}\right) + c \tag{32}$$

Now a more simple equation for distance can be generalised, using the distances of the external points to the line from the perpendicular slope:

$$D = \sqrt{\left(a - \frac{k - c}{m - m_2}\right)^2 + \left(b - \left(m\left(\frac{k - c}{m - m_2}\right) + c\right)\right)^2} \tag{33}$$

An additional equation for D using the other formulae for x and y can also be used, to remove the necessity of solving for the second line's constant:

$$D = \sqrt{\left(a - \frac{mb - mc + a}{1 + m^2}\right)^2 + \left(b - \frac{m^2 b - m^2 c + am}{1 + m^2} + c\right)^2} \tag{34}$$

To find the average of all of the distances of all external points to the calculated line, this process is repeated for every point, which can be expressed below:

$$A = \frac{1}{n_D}\sum_{i=1}^{n_D} D_i \tag{35}$$

D_i represents the ith point corresponding to its distance to the line, where n_D represents the number of points. The best line is defined as the line with the lowest average distance from external points to itself. The pair of points yielding the lowest average distance will be returned by the algorithm as the best line.

2.3 Using Linear Regression

Linear Regression is the faster and more accurate alternative to the previous method. Suppose a line in the following form, such that α and β are constants:

$$y = \alpha + \beta x \tag{36}$$

Using the data, both α and β can be calculated using the following formulae:

$$\beta = \frac{\sum_{i=0}^{n}\left(x_i - x_a\right)\left(y_i - y_a\right)}{\sum_{i=0}^{n}\left(x_i - x_a\right)^2} \tag{37}$$

$$\alpha = y_a - \beta \tag{38}$$

n represents the length of the dataset minus one, where x_i and y_i represent the ith index of their respective variable, and x_a and y_a representing the average of all of the x and y values respectively. A prediction function can be obtained significantly faster than the previous method, considering the $O\left(n\right)$ runtime of Linear Regression compared to $O\left(n^3\right)$.

3 Weight Calculations

3.1 Fundamental Logic

Before utilising the algorithms to calculate distance and approximate lines, the classes themselves need to be converted into numbers so that they can be manipulated in such a way. In the weight functions, the model calculates weights based on the loss and change values of previous iterations and ensures that weights are in increasing order. If previous scores or points do not exist, they will be assigned zeroes in the same format as if they had information. A multiplier is used to amplify the effects of changes conducted on new weights, where the global loss value must be equivalent to 0 for that to happen. Suppose the loss variable be expressed as l, where the function for general multiplier g_m is shown below:

$$g_m = \begin{cases} 1 & l \neq 0 \\ 50 & l = 0 \end{cases} \tag{39}$$

With each value of weights, individual calculations are done to determine their new values. In this case, consider a dictionary containing weight values with their respective accuracy scores from two previous iterations. Let p_{pi} and p_{ri} denote the ith weight's previous two iterations' accuracy scores, where individual loss is calculated below:

$$\Delta l = p_{pi} - p_{ri} \tag{40}$$

The change in weights can be done by taking ith weight from the two previous iterations:

$$\Delta w = p_{pwi} - p_{rwi} \tag{41}$$

With calculations done to obtain both Δl and Δw, new weights can be calculated. There are three cases for each possible pair of loss and change values.

3.2 Case 1: $\Delta l < 0$

As loss is negative, the accuracy of the previous model was decreasing. The model will use the average of the previous weight and the best weight with the same index, and subtract by the change in weights multiplied by a factor of g_m and the highest score, h_s. This enables for the new weight to contradict the current change in the weights, and attempt to slightly alter weight in a way which would result in positive loss. Subtraction is done, since change will be negated and will hence result in the contradiction of change from previous iterations. For example, if change is positive, there will be subtraction to contradict the detrimental relationship with loss that the positive change has, a similar case with negative change. The following formula is applied to conduct this operation:

$$n_w = \frac{(p_{pi} + b_{wi})}{2} - \frac{(\Delta w \cdot g_m)}{h_s} \tag{42}$$

3.3 Case 2: $\Delta l > 0$

As loss is positive, the accuracy of the previous model was increasing. The model will continue the positive trend by continuing to add change multiplied by g_m to the previous weight, using the following formula:

$$n_w = p_{pi} + \Delta w \cdot g_m \tag{43}$$

3.4 Case 3: $\Delta w = 0$ or $\Delta l = 0$

Without any change nor individual loss, the model will add the previous weight to the corresponding best weight of the same index multiplied by the highest percentage score, which is entirely multiplied by some random number in between 0.8 and 1 (shown from the *rand* function). The operation is conducted using the formula below:

$$n_w = (p_{pi} + b_{wi} \cdot h_s) \cdot \frac{rand(80, 100)}{100} \tag{44}$$

3.5 Generalising All Three Cases

To account for all three cases, the formula for the new weight n_w can be expressed as a piecewise function:

$$n_w = \begin{cases} \frac{(p_{pi}+b_{wi})}{2} - \frac{(\Delta w \cdot g_m)}{h_s} & \Delta l < 0 \\ p_{pi} + \Delta w \cdot g_m & \Delta l > 0 \\ (p_{pi} + b_{wi} \cdot h_s) \cdot \frac{rand(80,100)}{100} & \Delta w = 0 \text{ or } \Delta l = 0 \end{cases} \tag{45}$$

When calculated, the absolute value of these weights will be obtained to retain positive values. The weight values in the formula will correspond to the type of class that is being used.

3.6 Ensuring the Order of Weights

A condition for the weights is that they have to remain in ascending order. To ensure this, each weight past the first index will be checked to ensure that the previous weight is lower in value. If this is not the case, then values will be added to n_w to ensure such a case. In the case for IB weights, suppose that s is the new value of n_w, which is obtained using the following formula:

$$s = n_w + \sum_{n=0}^{k} \left((l + h_s) \left(\lfloor \frac{n}{100} \rfloor + 1 \right) + \frac{w_{i-1}}{n_w} \right) \tag{46}$$

k represents an integer, which would occur when s is greater than the previous calculated weight value. The maximum value for k is 9999, to prevent the algorithm from having too many iterations. The limit in iterations implies that

there will be a case where n_w will still be less than w_{i-1}. The algorithm will conduct the following operation to ensure that it will still be greater in value:

$$w_i = s + w_{i-1}\left(h_s + \left(\lfloor\frac{k}{100}\rfloor + 1\right)\right) \tag{47}$$

A similar operation is done for MYP weights, shown below:

$$s = n_w + \sum_{n=0}^{k}\left((l + h_s)\left(\lfloor\frac{n}{100}\rfloor + 1\right)\right) \tag{48}$$

With methods to calculate new weights, these now have to be combined into one general formula for the algorithm to use in the training process.

3.7 Generalising An Equation For Weights

To generalise a formula for ith IB weight, the formulae for n_w and s can be leveraged to do so. Suppose that S_c is a function to denote the sum to maintain ascending order in weight values for the specific class, c. The general formula for the ith weight for IB is below:

$$ib_{wi} = |n_{wi}| + S_{ib}\left(|n_{wi}|\right) + T\left(|n_{wi}| + S_{ib}(|n_w|)\right) \tag{49}$$

The sum functions utilise the same functions that are defined in Sec 3.6, where the IB sum function is below along with function T:

$$S_{ib}(x_i) = \begin{cases} 0 & i = 0 \text{ or } x_i \geq ib_{w(i-1)} \\ \sum_{n=0}^{k}\left((l + h_s)\left(\lfloor\frac{n}{100}\rfloor + 1\right) + \frac{ib_{w(i-1)}}{n_w}\right) & x_i < ib_{w(i-1)} \end{cases} \tag{50}$$

$$T(x_i) = \begin{cases} 0 & i = 0 \text{ or } x_i \geq ib_{w(i-1)} \\ ib_{w(i-1)}\left(h_s + \left(\lfloor\frac{k}{100}\rfloor + 1\right)\right) & x_i < ib_{w(i-1)} \end{cases} \tag{51}$$

The function T is used to ensure that ib_{wi} will be always be greater than $ib_{w(i-1)}$. A general equation for the ith MYP weight is very similar in structure to the IB function, however without function T:

$$myp_{wi} = |n_{wi}| + S_{myp}\left(|n_{wi}|\right) \tag{52}$$

The MYP sum function is very similar to the IB function, which also utilises the operation used to ensure order in Sec. 3.6 for MYP weights:

$$S_{myp}(x_i) = \begin{cases} 0 & i = 0 \text{ or } x_i \geq myp_{w(i-1)} \\ \sum_{n=0}^{k}\left((l + h_s)\left(\lfloor\frac{n}{100}\rfloor + 1\right)\right) & x_i < myp_{w(i-1)} \end{cases} \tag{53}$$

With equations to find the ith weight for both the IB and MYP, data can be properly converted into numeric values for proper predictions to be made by the algorithm.

4 Training the Model

4.1 Generating Weights

To allow for proper predictions to be made, the model first needs to be trained. There is a given amount of iterations done to run the algorithm, which will find the best model within that set of iterations. If the algorithm is in its first iteration, a default set of weights are assigned since there have not been any previous iterations. In the actual implementation of the algorithm, the default IB weights are $\{0, 1, 2, 3\}$, where the MYP weights are $\{0, 1, 2, 3, 4, 5, 6\}$. In further iterations, previous weight values exist, meaning that the algorithm can train an accurate model. In this case, class weights are generated.

$$ib_w = \{ib_{wi} | 1 \leq i \leq 4\} \tag{54}$$

$$myp_w = \{myp_{wi} | 1 \leq i \leq 7\} \tag{55}$$

Subsequent dictionaries containing classes and their corresponding weight values are generated, which will aid in finding weight values and reversing them when making predictions. A general format of the weights dictionaries are below:

$$\begin{aligned} ib_d = \{&\text{"AISL"}: ib_{w1}, \text{"AIHL"}: ib_{w2}, \text{"AASL"}: ib_{w3}, \text{"AAHL"}: ib_{w4}\} \\ myp_d = \{&\text{"Geo F."}: myp_{w1}, \text{"Geo"}: myp_{w2}, \text{"Alg 2 T. F."}: myp_{w3}, \text{"Alg 2 T."}: \\ &myp_{w4}, \text{"Alg 2 T. H."}: myp_{w5}, \text{"AAC1"}: myp_{w6}, \text{"AAC1 H"}: myp_{w7}\} \end{aligned} \tag{56}$$

A reverse IB dictionary is generated to take weight input and return a class, which is simply the inverted version of ib_d.

4.2 Parsing Data and Predicting Linear Relationships

Weight generation is followed by data parsing to be able to manipulate given data and use the weights to find a linear relationship between them. The format of class data is shown below, which will entail how data will be parsed for usage in training:

$$MYP : \text{Name, Graduation Year, Math Level} \tag{57}$$

$$IB : \text{Name, AI Level, AA Level} \tag{58}$$

In the IB, a student can only take one type of maths course, which means that one of the class columns will be blank. The column which is not blank is the class that the student is taking, where the data indicates their level. If the data seems to be missing this information, the student will be added to a missing persons array, which will be used when predicting the classes of these students. For students who are not missing, the weight values of their current class and IB class will be added to an array containing the statistic of known class routes. When completed, the best line is calculated using the algorithm and theory from Sec. 2.

4.3 Making Predictions

Once obtained, the linear function can be used to make predictions. Suppose that the raw prediction function (the calculated line) is $f(w)$, such that w is an MYP weight. Function f will return a value, where the closest IB weight to it will be used as the prediction. Suppose that the raw weight be defined as p_{raw}, which is the output of $f(w)$:

$$p_{raw} = f(w) \tag{59}$$

There are two weights that are obtained as a result of the operation, which will then obtain the high and low predictions:

$$p_{high} = \lceil p_{raw} \rceil \tag{60}$$

$$p_{low} = \lfloor p_{raw} \rfloor \tag{61}$$

Now, to find the distance of prediction p (applying to both p_{high} and p_{low}) to the ith IB weight:

$$D = |p - ib_{wi}| \tag{62}$$

The weight yielding the smallest value of D is the prediction made by the model. To widen the scope of predictions, the algorithm will take the floor and ceiling of the value returned by the prediction function and make high and low predictions based on the new values.

4.4 Assessing the Accuracy of Models

When training, the obtained model will be able to make predictions, however such predictions will vary in accuracy. To ensure that the obtained model is accurate, a test is ran to assess its accuracy. The algorithm uses the existing data obtained from data parsing, where the calculated line is used to predict the classes of students based on their MYP weight value. If their predicted class matches the actual class, the model has made a correct prediction. Eventually, the accuracy is obtained when iterating through every known data set, where accuracy is returned as a percentage score. In each MYP class, there are individual points that are assigned which gets the accuracy score for predictions in the specific class, which will be used in weight calculations. When accuracy is obtained, the global loss is calculated using the obtained score and the previous. Suppose that P is the current score and P_p the previous.

$$l = \Delta P = P - P_p \tag{63}$$

The loss value will be used in weight calculations. Once completed, the currently obtained values will be mapped to previous weight values for proceeding iterations, as well as previous weight values being mapped to prior values. If the obtained score happens be larger than the high score, the algorithm use the current values as the final model's values. This process is repeated for the set amount of iterations given by the user.

5 Extra Information

5.1 Assessing the Probability of Predictions

In the implementation, a probability score is attached to predicted values. To obtain this probability score, the algorithm assesses the IB classes that are a result of previous MYP classes. Since the model makes two predictions, it can be reasonably inferred that the probability of the two classes are mutually exclusive and have a total probability of 1. This can be represented using probability. Suppose that x is the high prediction, where y is the low. Let function P represent the probability of its argument.

$$P(x \cup y) = P(x) + P(y) = 1 \tag{64}$$

Hence, the equation can be rearranged to obtain $P(y)$ in terms of $P(x)$:

$$P(y) = 1 - P(x) \tag{65}$$

If the probability function obtains more than two separate values with probabilities above 0, the equation for $P(y)$ will ensure that the probabilities of both predictions will always add up to 1.

5.2 Python Implementation and Test Data

The direct Python Implementation of the algorithm can be downloaded **here**. General MYP data can be downloading by going **here**, where IB data is provided using this **link**.