1. **Method Overview**:

   I'm using kNN (k- nearest neighbors) for classification. I have taken help from scikit-learn (sklearn)'s nearest neighbors module to perform classification. To ensure transfer learning my classifier assigns '+' class to only those rows which are among the 20% of the positive class. For this, I've used inbuilt predict_probability() function from the library and used some math.

   Additionally, I've used manual feature selection to get my results. I'm using all the features except the nominal features and 'Sex' feature.

   The decision to choose these was based on my assumption that above-mentioned features are primary contributors to students' performance and their ability to use the resources at an optimum level. I have removed features like 'parents' jobs' (mother and father) as I think this should never be a determinant of a students' performance. Also, other features I've removed are on my sole discretion that they shouldn't predict student performance.

   I also wanted to mention that I knowingly removed 'Sex' feature to not make my classifier biased. Other nominal/categorical variables have also been removed for the similar reasons stated above. Besides, kNN does not perform well on the categorical variables which has also contributed in my decisions for feature selection.

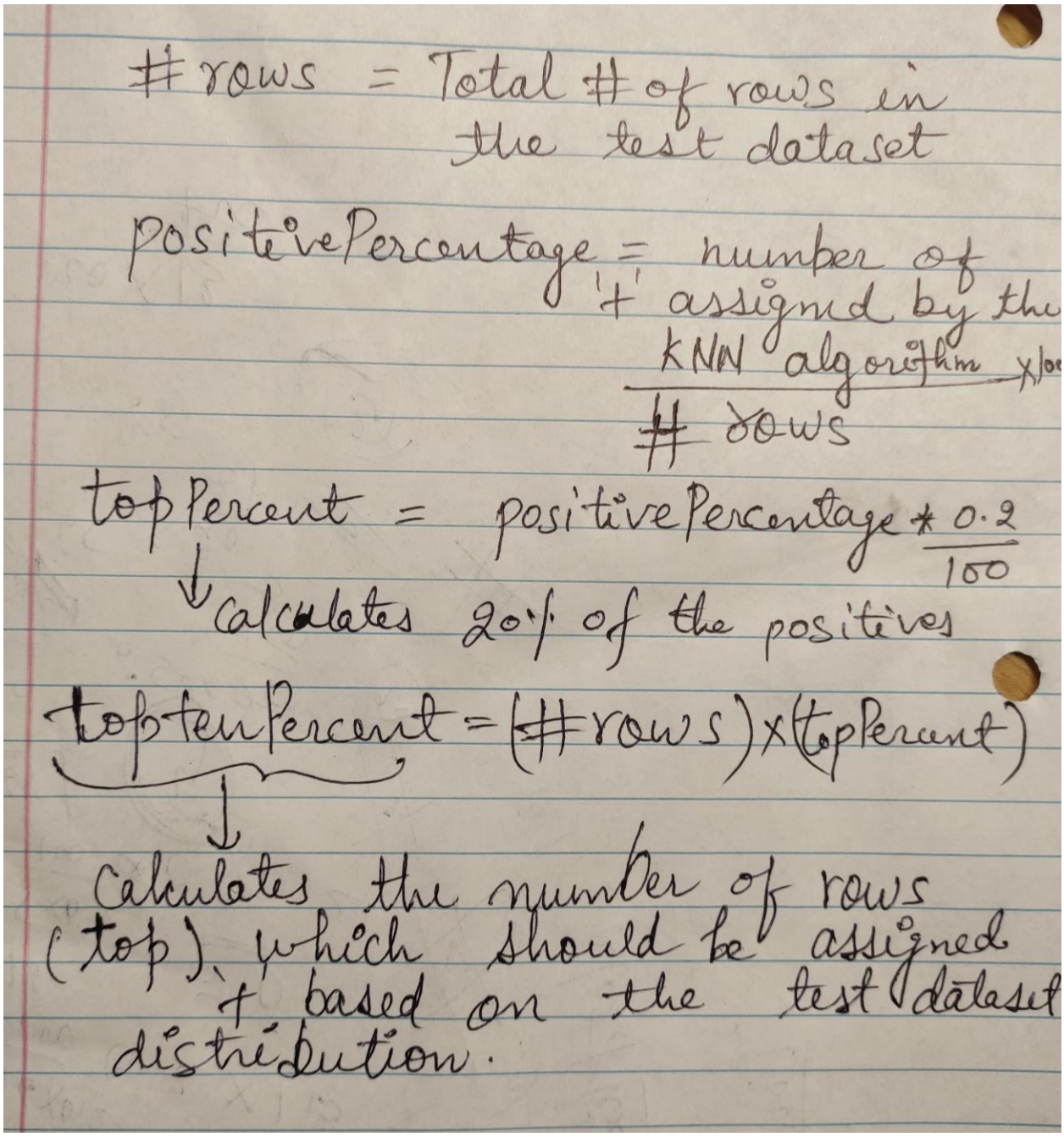
   Furthermore, I've used Min-Max Scaling to make sure that one feature does not control the results of the algorithm.

2. **Constraint Satisfaction**:

   My approach is using the predict_probability() function to obtain the probability for the class of each row in the test dataset. Further, from the probabilities of each class, I obtain the indices of the topmost 20% (in terms of probability) rows which have been assigned a positive class already.

   This makes sure that the resources are expected to return 10% positives as required by the assignment.

   Below is the proper math idea, I included in my code to satisfy the constraints:



Below is the mathematical formula used for Min-Max scaling:

Min-Max scaling:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

I used the sklearn library for scaling.

3. **Efficacy:**

My method returns the best 10% individuals because I am sorting the predicted classes based on their probability and then I'm only assigning a '+' class to the test row if it is among the top 20% probability rows of the positives classes already predicted by the kNN. This ensures that the best resources go to no more than the best 10% individuals in the dataset.

The algorithm that I've used using kNN to find the distribution of '+' and '-'classes in the dataset. Further, no more than the 20% of the positive classes are assigned a '+' class. This makes sure that if the kNN predicts all '-' in the dataset than the top 20% of the positives is 0 and no row in the dataset is assigned a '+' positive class and no one is included in the best 10% of the individuals.

Additionally, 20% of 50 (positives) is 10% of the total 100 individual for a dataset. This idea is eccentric to my algorithm to work efficiently.

4. **Table:**

Some of the tests were taken from data.csv using test_train split module in sklearn. I took 80% for training and 20% for testing from data.csv.

Also, I've tested for values of k (number of nearest neighbors to run the algorithm on) from 9 to 41 and found k=29 to be somewhat working better than other k values and all the tests below were conducted with k = 29.

Besides, the distance metric used for all the tests is Euclidean distance metric.

The table has some of the tests I used to empirically test my kNN classifier.

No of rows in the test file (Excluding the header)	**Assigned '+' by the kNN algorithm originally (Used to find distribution in the dataset)**	**Assigned '+' based on the top 20% of the positive class (Printed class)**	**Remaining number: classes are printed/assigned '-'**	**Comments**
5	**1**	**0**	**5**	**Test manually taken from data.csv**
13	**3**	**0**	**13**	**Test dataset with all negatives**

				taken from data.csv
100	**75**	**15**	**75**	
100	**70**	**14**	**86**	
136	101	20	116	Random test taken manually from data.csv
100	**67**	**13**	**87**	**Accuracy from kNN = 76%**
10	**10**	**2**	**8**	**Test dataset with all positives (100% - 0 % distribution for '+' and '-' repectively)**

References:

1. http://sebastianraschka.com/Articles/2014_about_feature_scaling.html#standardization-and-min-max-scaling
2. https://www.python-course.eu/k_nearest_neighbor_classifier.php#Voting-to-get-a-Single-Result