HW1: Data Exploration and Preparation

By: Hani Azzam & Naser AlBasar

Part 1: Data Loading and First Look

(Q1) There are 1250 rows and 26 columns.

(Q2) We think this feature refers to how many conversations the subject had per day on average.

```
1     232
2     213
3     191
4     158
5     127
0     117
6      69
7      46
8      34
9      21
10     20
12      7
11      5
13      4
15      3
14      2
16      1
Name: conversations_per_day, dtype: int64
```

The choice to make this feature's type "ordinal" stems from the difficulty of defining what constitutes a 'single full conversation' or a partial one on a continuous domain. It is therefore better to think of it as somewhere in between ie 'Ordinal'.

(Q3)

Feature Name	Description	Type
patient_id	Id of patient	Other
age	Age of of patient	Ordinal
sex	Sex of patient	Categorical
weight	Weight of patient	Continuous
blood_type	Blood type of patient	Categorical
current_location	Current location of patient	Other (2D Vector of continuous

		variables)
num_of_siblings	Number of siblings patient has	Categorical
happiness_score	How happy patient feels	Ordinal
household_income	Patient's household income	Ordinal
conversations_per_day	[See Q2]	Ordinal
sugar_levels	Sugar level in patient blood	Ordinal
sport_activity	Patient's level of sport activity	Ordinal
symptoms	Patient's symptoms	Other
pcr_date	Date of PCR test	Other
PCR_01 … PCR_10	PCR test 1 throw 10	Continuous

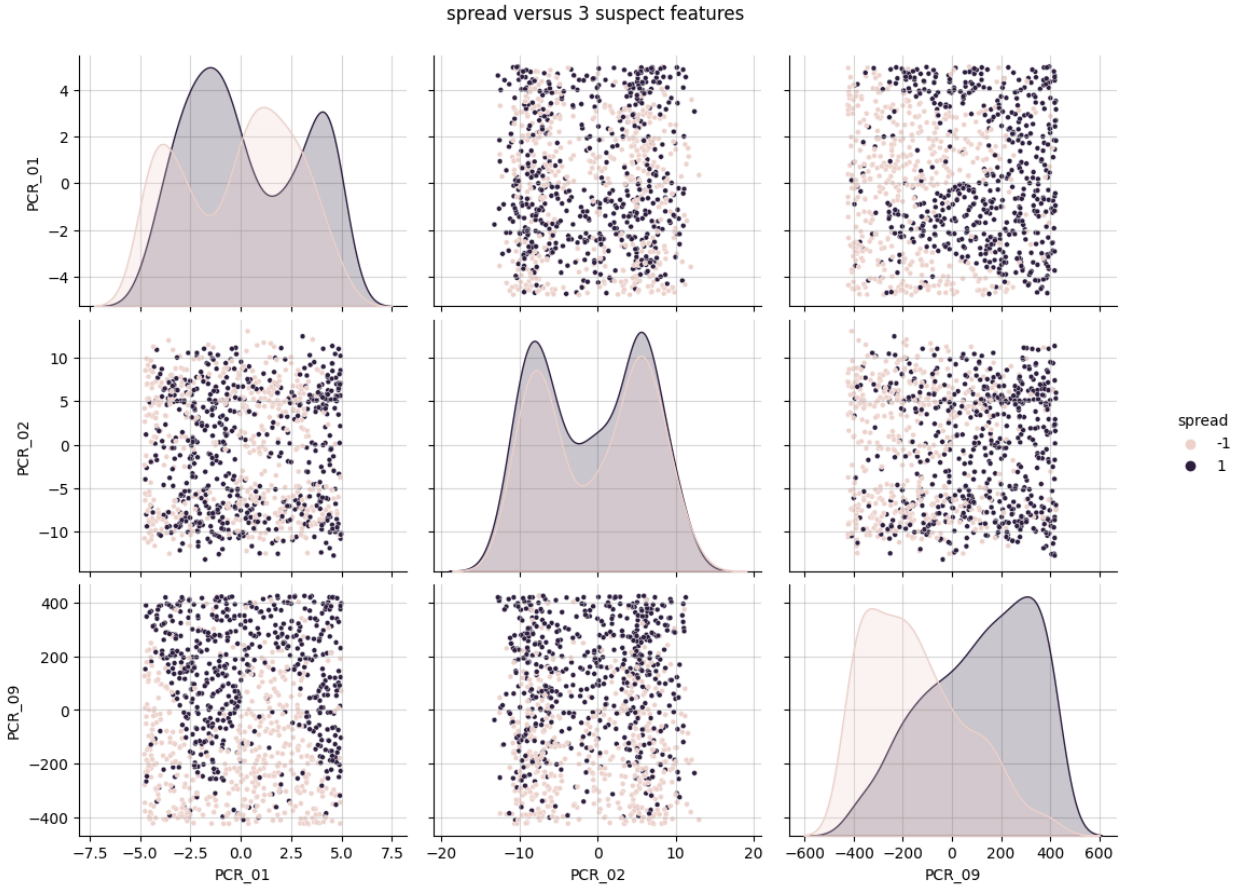
(Q4) We use the exact same split for all our analyses, because we are interested in examining how different methods of data manipulation on the training set affect the machine learning process. Checking the effects of different splits is not an interest of ours, therefore it's important to keep it fixed.

Part 2: Warming up with k-Nearest Neighbors

(Q5)

```
Correlation between spread and PCR_01 is: 0.080
Correlation between spread and PCR_02 is: -0.028
Correlation between spread and PCR_09 is: 0.523
```

(Q6)



PCR_01 and PCR_09 are the most useful to predict the spread because PCR_02 is non separable and we cannot learn anything from it.

(Q7)

1.calculating the distance from all points

1.1 each calculation is O(d)

-> O(d*N)

2. Partition: O(N)

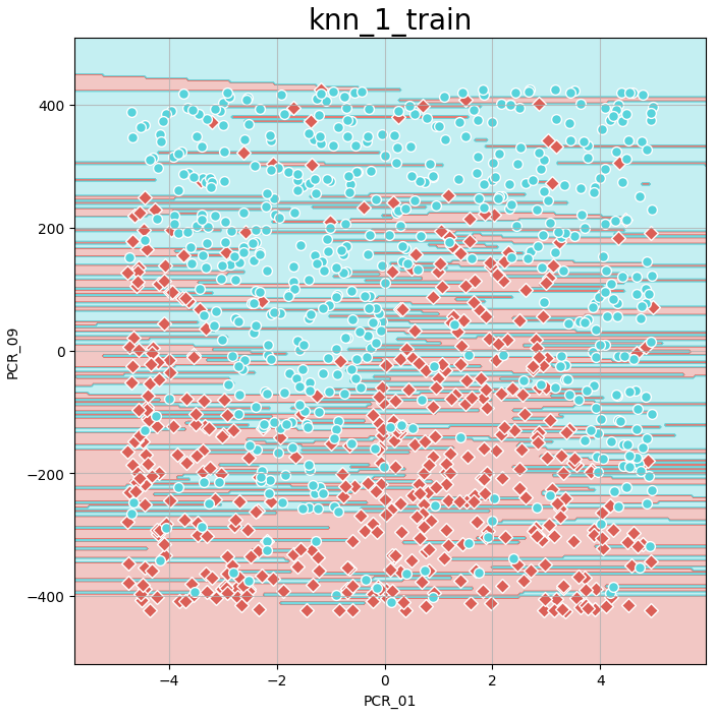
3.get the k neighbors label: O(k)

k<=N o.w. K is obsolete

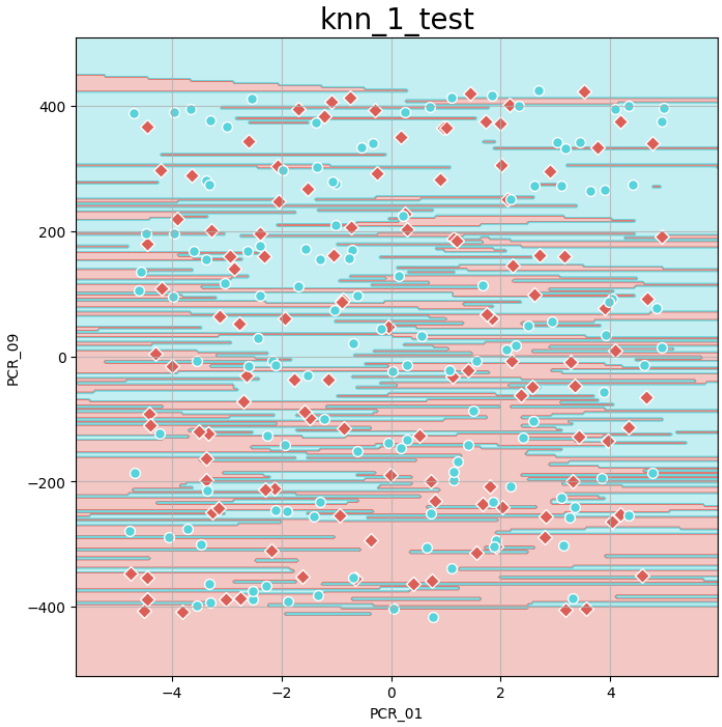
Overall: O(d*N)

(Q8)

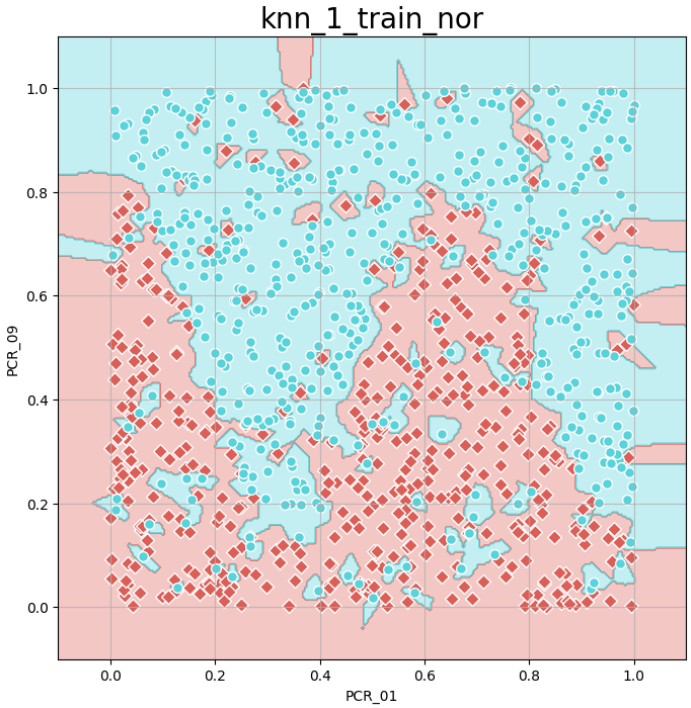
Accuracy on training set: 1.0



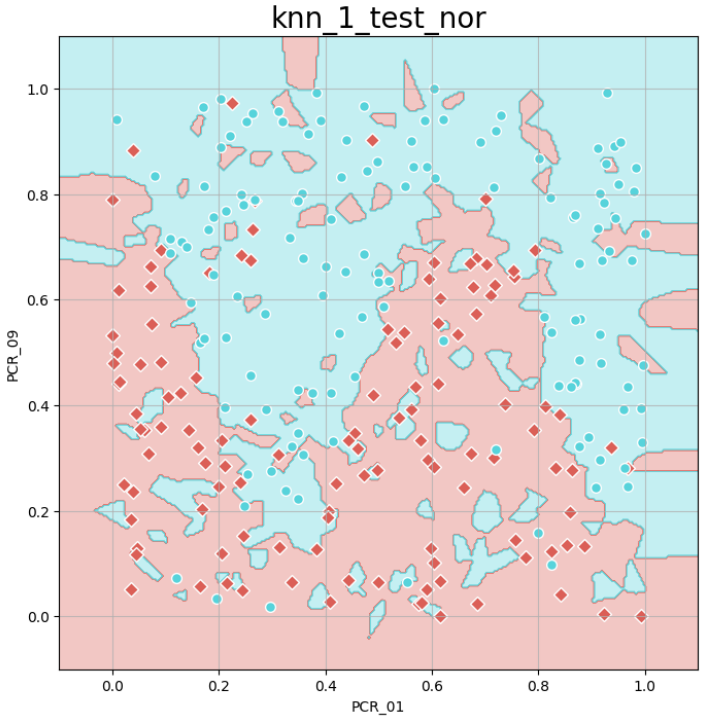
Accuracy on test set: 0.716



(Q9)



Accuracy on training set: 1.0

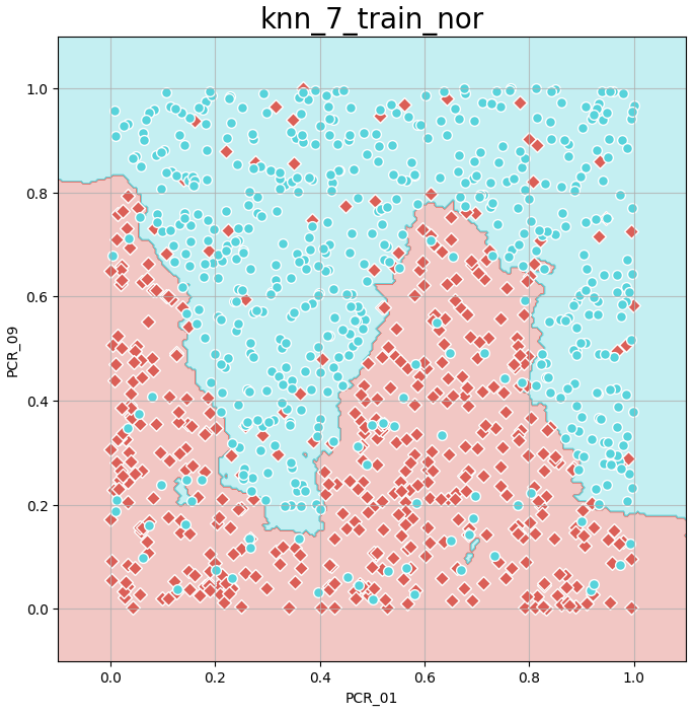


Accuracy on test set: 0.784

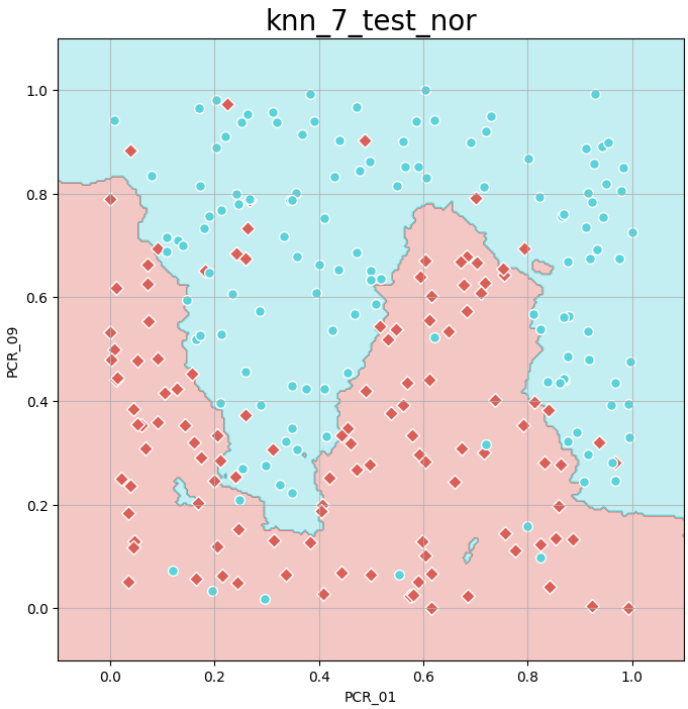
The KNN model relies on measuring euclidean distances of a given sample to the samples in the train set, then labeling it based on the label of K nearest samples. If we measure this

distance on 2 axes; x and y, where axis x, has a range that is considerably smaller than the y axis (as is the case with PCR_1 existing on range of ~[-5,5] which is much smaller than PCR_9's ~[-200,200]), then the nearest neighbors to any given sample will be mostly dictated by the smaller axis, because its x-distance would be far smaller than the y-distance, meaning the y axis gets heavily neglected. Therefore, it is important to normalize the axes accordingly, to ensure that neither gets neglected.
Here is an example: Given axis x with range of [0,1], and axis y of range [1, 100], And the following training set:

(Q10)



Accuracy of training set: 0.882



Accuracy of test set: 0.872

Our accuracy went up when we took more than one neighbor into consideration, i.e. we aren't overfitting anymore

(Q11)

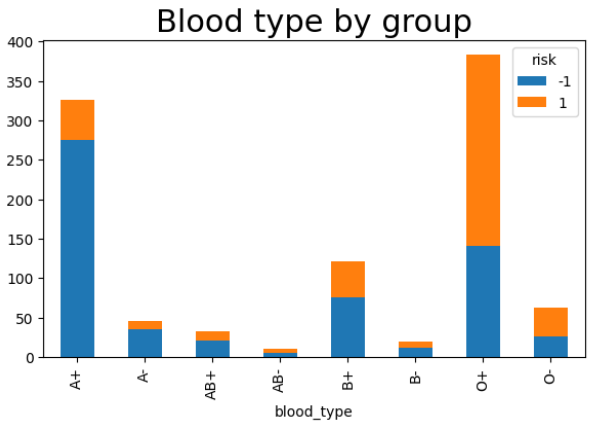
Since chi-squared is unbound we cannot normalize the data set given by it since it doesn't have a max value. When we normalizing our data with min-max we will “over-normalize” them to the point of data loss, by suppressing most of the data points, which are closer to 0, in favor of the rest of the data points which are sparsely distributed across the higher values leading up to the max value. Normalizing the uniform data will only shift it to the left and compress them a bit.

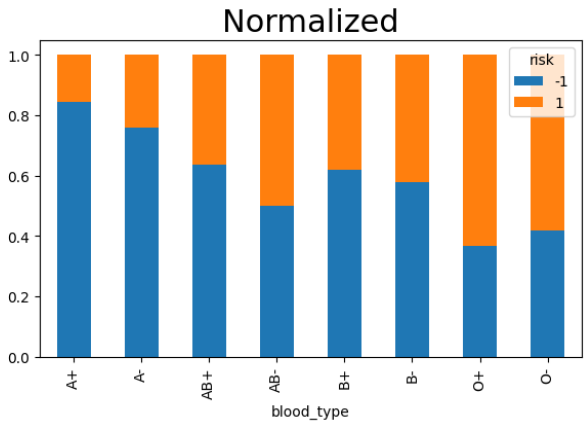
Part 3: Data Exploration:

(Q12)

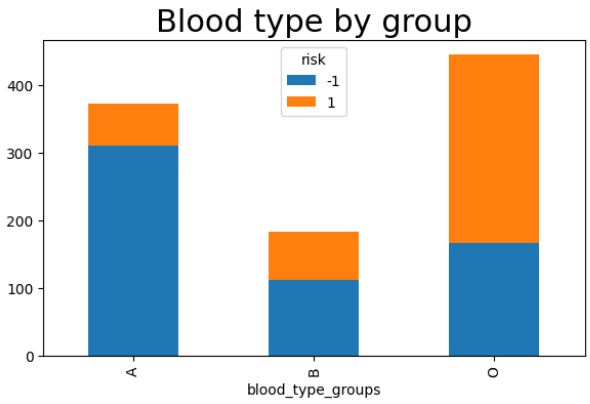
we will need 5 boolean values, the one for each category and one for the sign, i.e: (“SIGN”)X(“A”, “B”, “AB”,”O”) : (1,0)X(1000,0100,0010,0001)

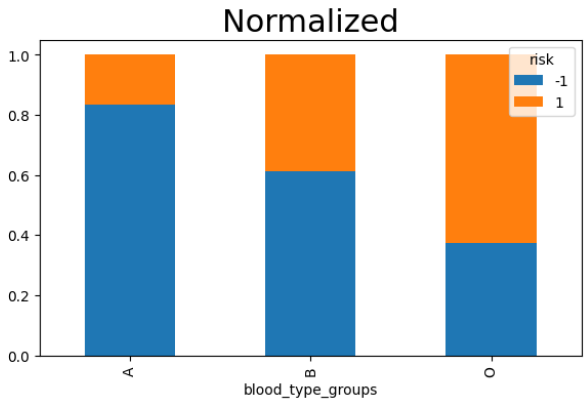
(Q13)





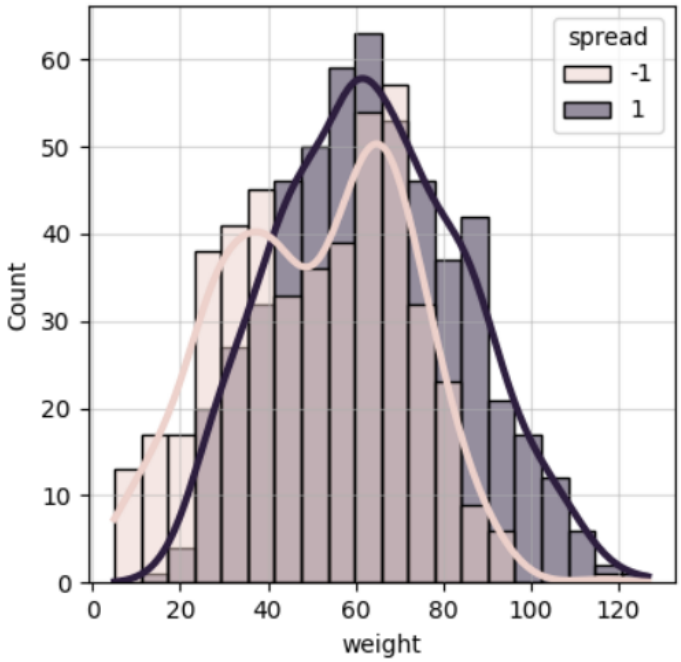
Since the ratio of risk=1 and risk=-1 in A+ is similar to that of A- we can bunch them up together and ignore the minor difference between them, same for AB+, AB-,B+ and B- and same for O+ and O-, this way we get to ignore the signs of each type and discard a feature.



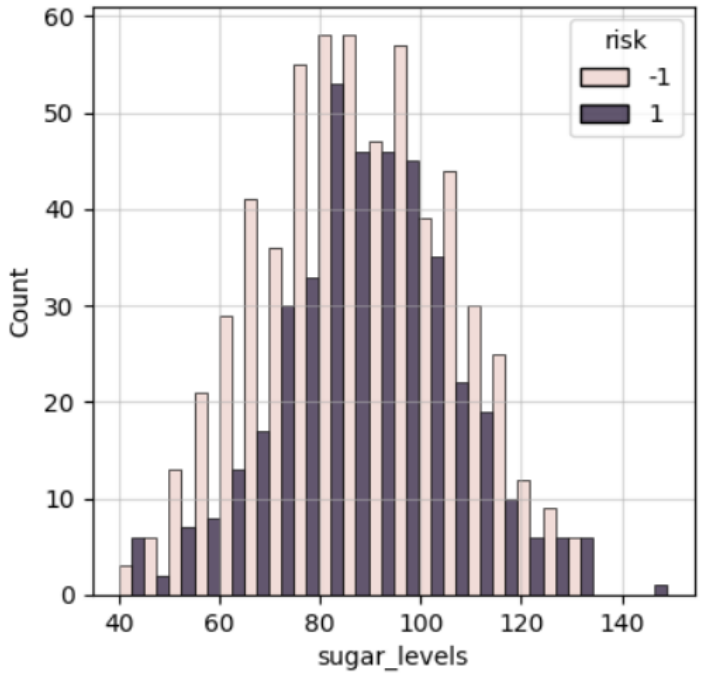


(Q14)
Instead of holding a string describing what symptoms each patient had, we transform this feature into 5 different boolean features whereas each feature describes if the patient suffered the symptom or not, we do this by iterating of each patient and checking what symptoms he suffered from, for each symptom he suffered we mark 1 in the corresponding symptom array and 0 if he didn't suffer from it.

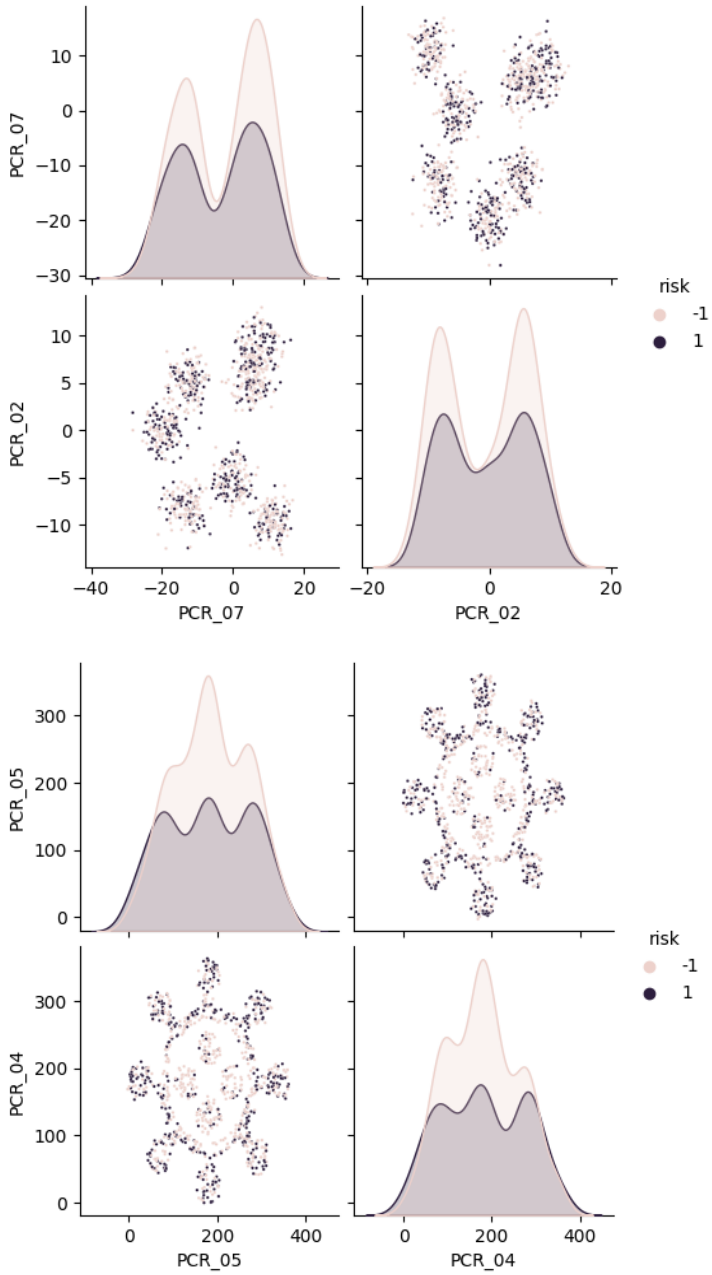
(Q15)
The most informative feature for predicting the *spread* target variable is *Weight*. This is because we are searching for a plot with the clearest difference between the -1 curve and 1 curve, which means the feature is the best predictor for a patient's spread, and that feature happens to be *Weight* (if it's high, the spread is probably -1, otherwise it's 1).



(Q16)
The most informative feature for predicting the *risk* target variable is *sugar_level*, same reasoning as above
That is because we can see a clear difference in the ratio between the 2 categories for each sub group.

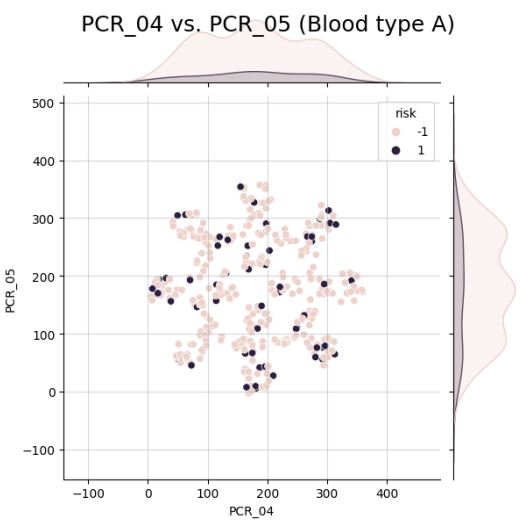

risk
-1
1
Count
sugar_levels
0
10
20
30
40
50
60
40
60
80
100
120
140

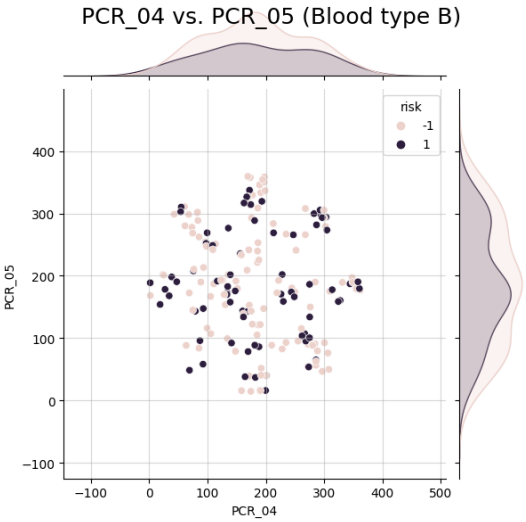
(Q17)

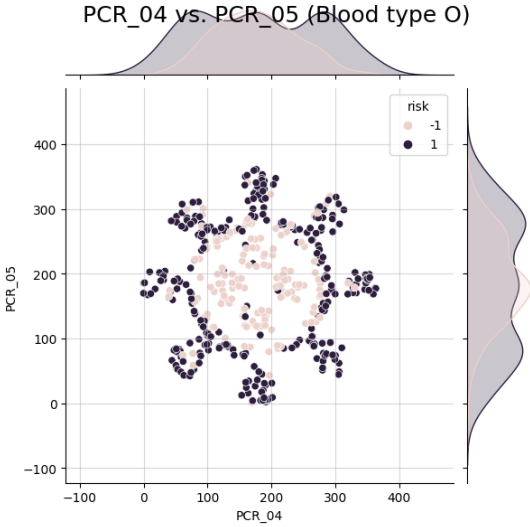


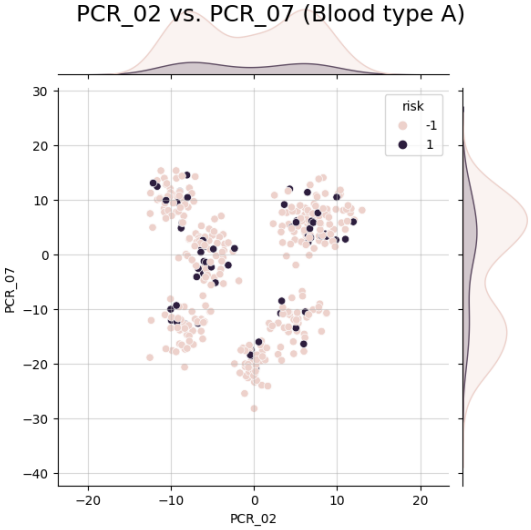
Based on the second plot we could infer that a point that belongs in the range of the 4 middle circles would likely get a risk value of -1, outside them, the data largely overlaps and no discernable decisions can be made.

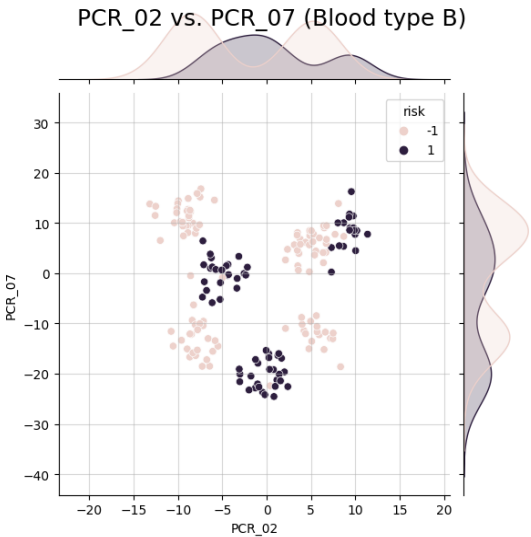
(Q18)

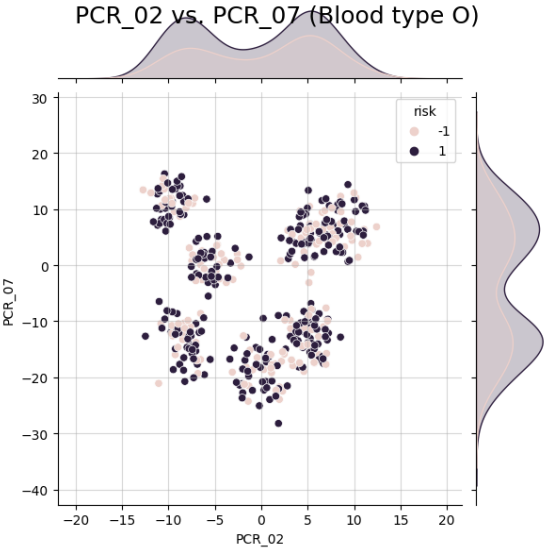

PCR_04 vs. PCR_05 (Blood type A)
risk
-1
1
PCR_05
PCR_04


PCR_04 vs. PCR_05 (Blood type B)
risk
-1
1
PCR_05
PCR_04


PCR_04 vs. PCR_05 (Blood type O)
risk
-1
1
PCR_05
PCR_04


PCR_02 vs. PCR_07 (Blood type A)
risk
-1
1
PCR_07
PCR_02


PCR_02 vs. PCR_07 (Blood type B)
risk
-1
1
PCR_07
PCR_02


PCR_02 vs. PCR_07 (Blood type O)
risk
-1
1
PCR_07
PCR_02

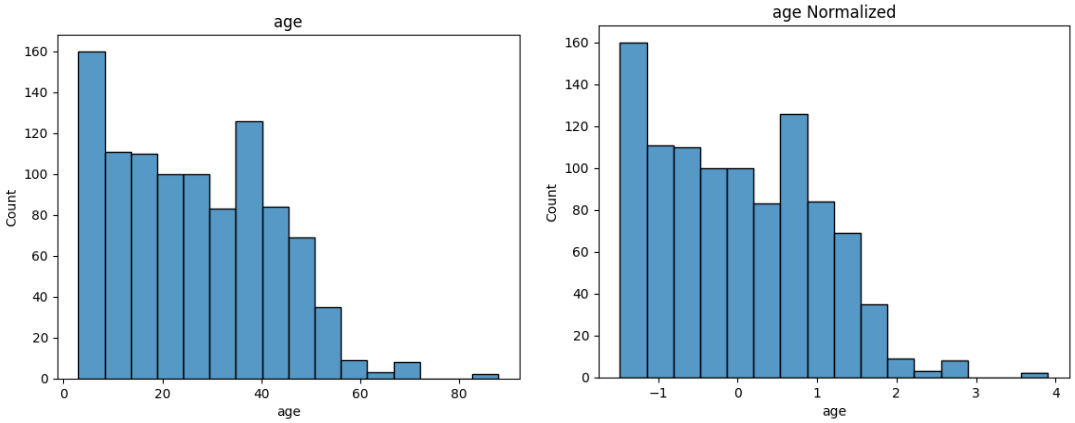
(Q19)

kNN: We expect this model to work very well when using features PCR_02 and PCR_07, with patients who have blood types ±B, because as we can see from its respective plot, the points of label '1' lie in 4 near-homogeneous clusters which are clearly separated from the 3 clusters of '-1' labels. As for the rest of the features, some may work, but to a much lesser extent.

Decision trees: we expect this method to work, if we do a decision tree that sorts based on blood type and then gives a prediction based on plot #3 from Q18 and plot #5 from Q18 and give a prediction of 1 for all samples with blood type A.

Linear models (with no mappings): won't work, because our data doesn't follow any linear patterns, therefore we cant pass one line that will be able to separate our data and be able to predict the risk of each data point.

(Q20)



(Q21)

Denote the features in the dataset as $\{f_1, f_2, \ldots. f\}$

(a) Forward feature selection trains d_1 models on its first step, the d_1-1 models on its second step and so forth, we get:

$$\sum_{i=1}^{d_2} d_1 - i + 1 = O\left(d_1 d_2\right)$$

(b) Similarly, backward feature selection first trains on d_1 models in the first step, each containing all the features but 1. In the second step, it trains d_1-1 models, each containing all features, except 2 (which include the one discarded in the last step). And so forth, hence, we get the exact number of the models we got in (a):

$$\sum_{i=1}^{d_2} d_1 - i + 1 = O(d_1 d_2)$$

(Q22)

Sample	x[1]	x[2]	x[3]	y
#1	1	1	-1	1
#2	1	-1	1	1
#3	1	-1	1	1
#4	-1	1	-1	-1
#5	-1	-1	1	-1
#6	-1	-1	1	-1

In forward selection, x[1] will be selected.
In backward selection, x[2] or x[3] will be selected.

(Q23) The classifier choose weight and PCR_01 and PCR_09, 2 of which we choose in Q6 and the third is the feature we choose in Q15

(Q24) It is important to normalize our data before doing this step because kNN is sensitive to the scale of our inputs therefore we must normalize our inputs for the purpose of not getting a skewed result that favours small scale features, this changes from learning algorithm to another like in the case of decision trees, which are not sensitive to the scale of the features.

(Q25)
Reasoning for normalization method after table

Feature Name	keep	new	Normalization method	explanation
patient_id	v	x	-	-
age	v	x	standard	-
sex	v	x	-	-
weight	v	x	standard	-
blood_type	x	x	-	

blood_type_groups	v	v	-	Bunched A, B and AB, O into groups and gave them numerical values A : -1 O : 0 B : 1
current_location	v	x	-	-
num_of_siblings	v	x	-	-
happiness_score	v	x	standard	-
household_income	v	x	standard	-
conversations_per_day	v	x	standard	-
sugar_levels	v	x	standard	-
sport_activity	v	x	standard	-
symptoms	x	x	-	Categorized the different symptoms into their own features
Smell loss	v	v	-	Based on symptoms, if suffered from smell loss marked 1 otherwise marked with 0
fever	v	v	-	Same as smell loss
cough	v	v	-	Same as smell loss
Shortness of breath	v	v	-	Same as smell loss
Sore throat	v	v	-	Same as smell loss

pcr_date	v	x	-	
normalized_pcr_date	v	v	MinMax	Normalized to days since starting of testing for corona
PCR_01 … PCR_10	v	x	standard	

Reasoning for standard normalization was to conserve features that had outliers such that they wont get too congested and affected by such outliers.

Otherwise min_max is preferable to conserve the shape of the data.