Machine Learning Major HW1:
Data Exploration and Preparation

By: Hani Azzam and Hamza Odeh

PART 1:

(Q1)
(1250, 26)
That is: 1250 rows and 26 columns.

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

```
conversations_per_day
3     224
2     215
4     190
5     156
6     111
1     104
8      72
7      60
9      39
10     23
11     19
12     12
13      9
14      6
17      4
15      2
16      2
19      1
22      1
Name: count, 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 through 10	Continuous

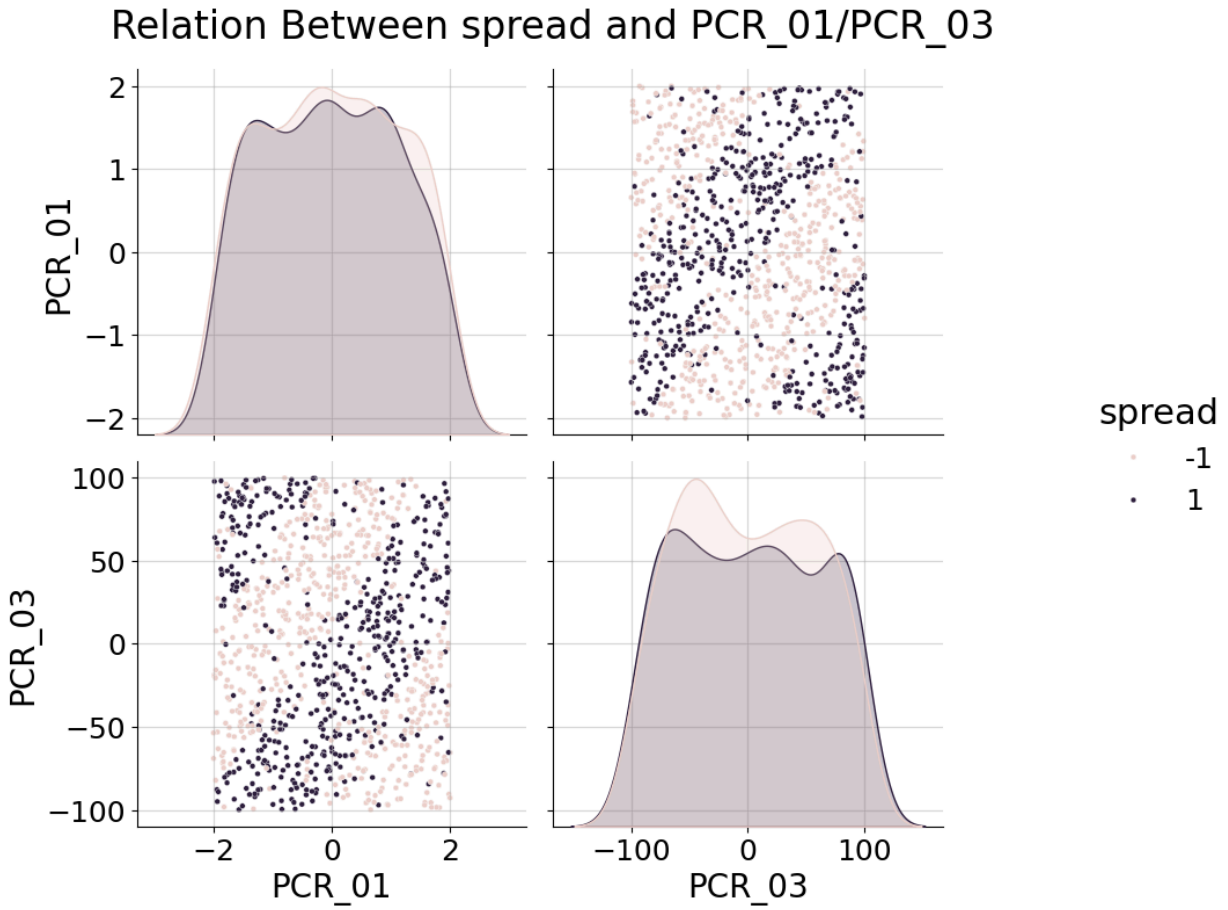
(Q4)

We use the same data split throughout our analysis for 3 main reasons:

1. Consistency: It ensures that the performance metrics are consistent and comparable across different models or algorithms. If the split changes, the problem could change, leading to inconsistent results.
2. Fair Comparison: When comparing different models, it's important that they are evaluated on the same grounds. Using the same split ensures that all models are tested on the same unseen data, making the comparison fair.
3. Reproducibility: Using the same split allows others to reproduce your results.

PART 2: Warming up with k-Nearest Neighbors

(Q5)



Yes, the features are useful We can see from the pair plot that the spread label is separable.

(Q6)

```
Correlation between spread and PCR_01 is: -0.011
Correlation between spread and PCR_03 is: 0.015
```

On their own, each of the 2 features has no correlation with the spread (within margin of error). But from what was observed in the plots in Q5, multiple correlation between the spread and the 2 features together is high. This suggests that PCR_01 and PCR_03 together have a significant multiple correlation with spread, but individually they do not.

(Q7)

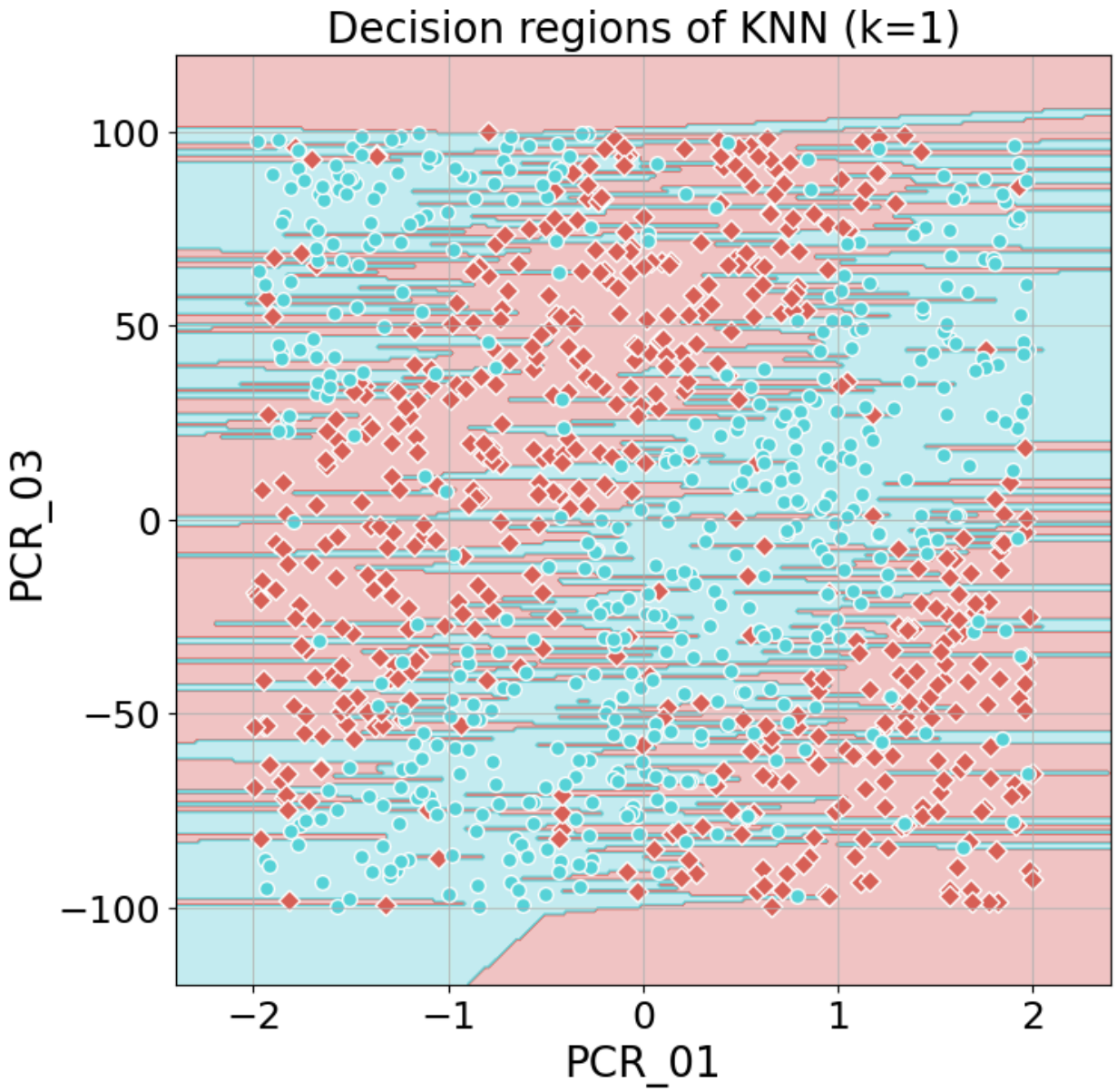
Time complexity:

```
def predict(self, X):
        distances = distance.cdist(X, self.X, 'euclidean') - O(m*d)
        k_nearest = np.argpartition(distances, self.n_neighbors)[:,
:self.n_neighbors] - O(m)
        predictions = np.sign(np.mean(self.y[k_nearest], axis=1)) - O(m)
        return predictions
```

O(m*d + m + m)

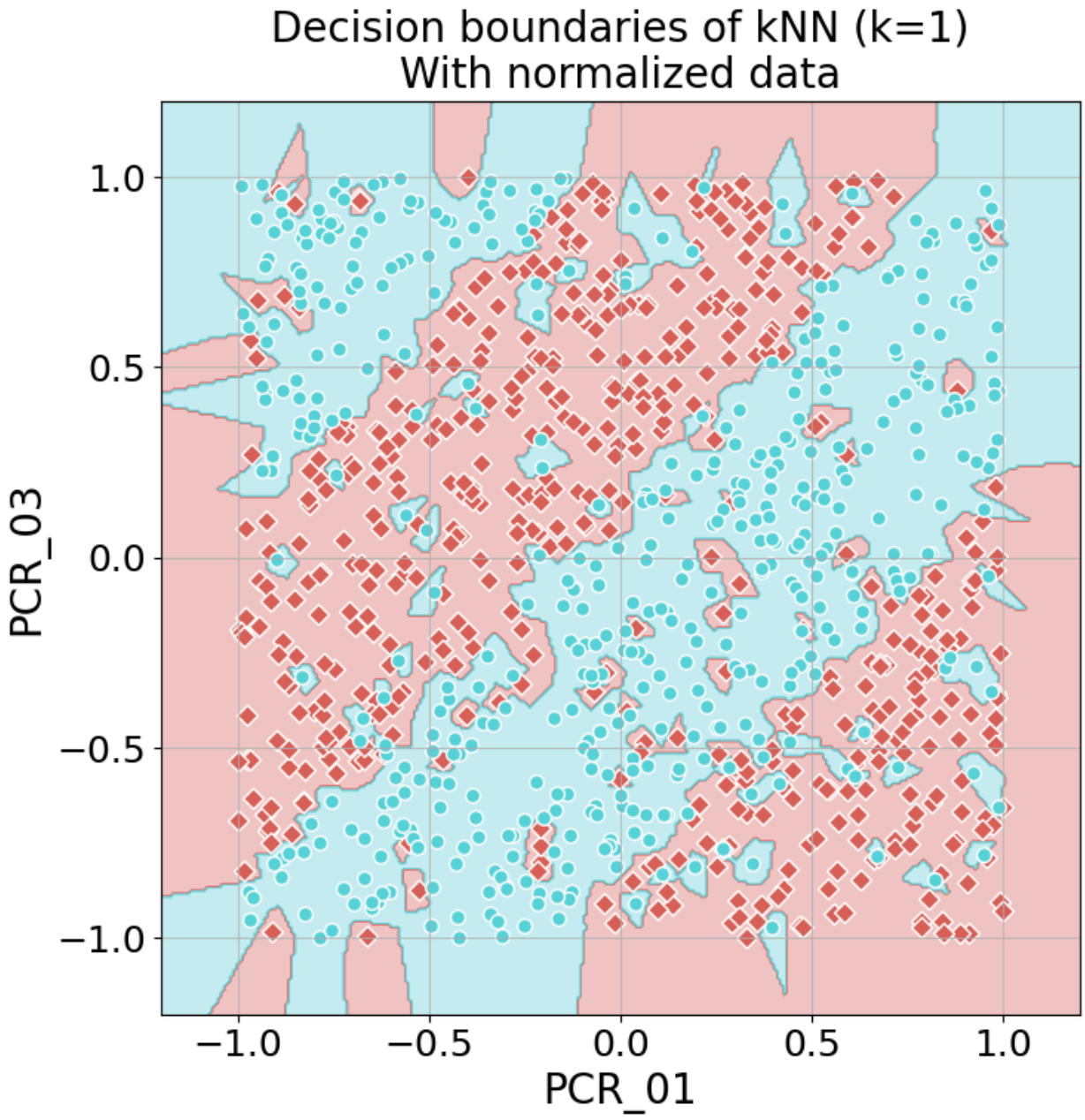
We get a total time complexity of O(m*d)

(Q8)




```
train accuracy:  1.0
test accuracy:  0.664
```

(Q9)

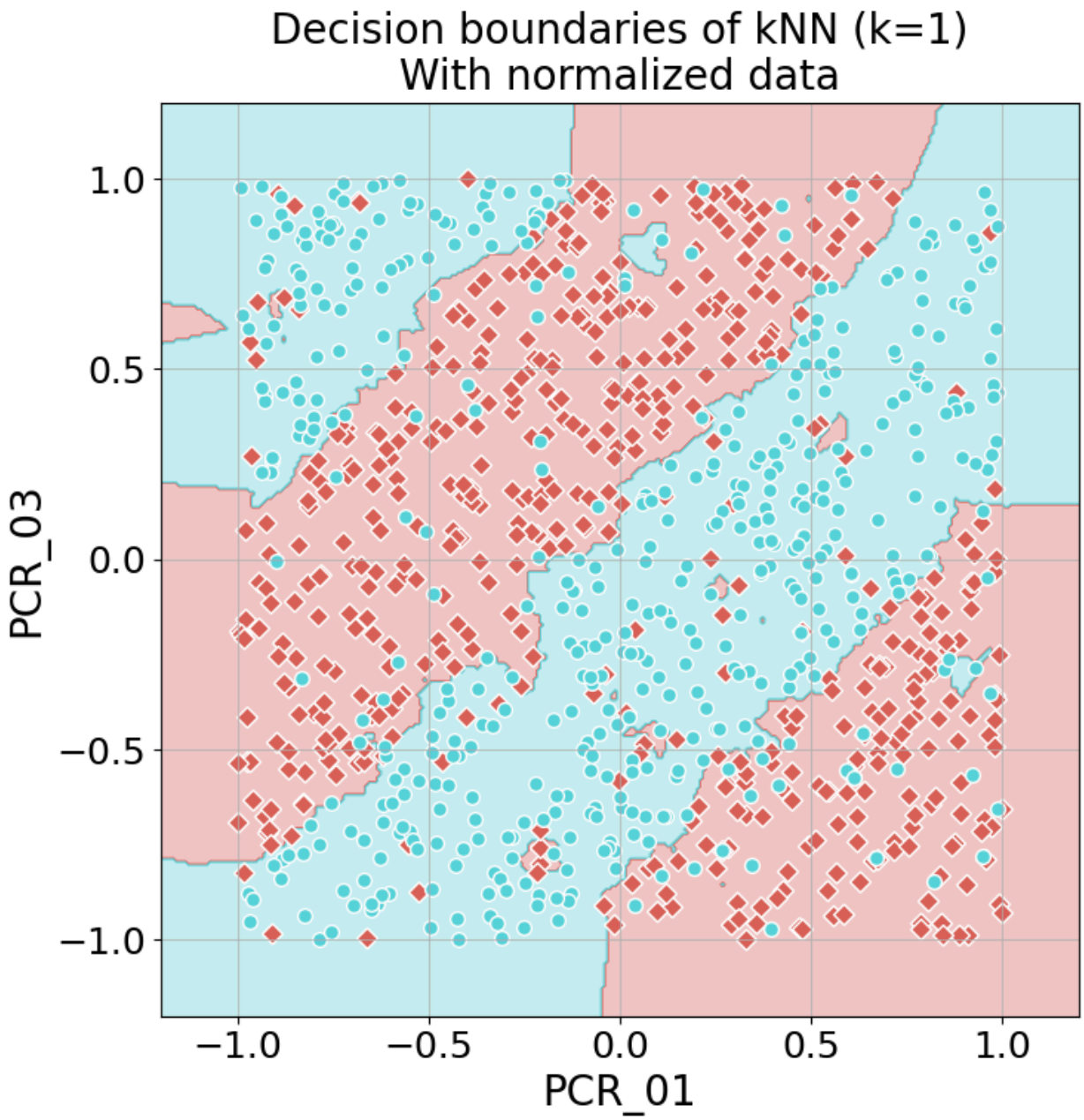


train accuracy: 1.0
test accuracy: 0.8

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_01 existing on range of ~[-1,1] which is much smaller than PCR_03's ~[-100,100]), 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.

(Q10)



train accuracy: 0.877
test accuracy: 0.86

In Q9 we used a lower k value (k = 1) which resulted in overfitting as can be seen in the test accuracy (0.8) compared to when we used k = 5 which resulted in test accuracy of (0.86) despite having a lower train accuracy (0.877) compared to k = 1 which has a perfect train accuracy. This was expected since using a higher k value (k = 5) results in a more generalized decision since we use the average of more neighbors compared to only one neighbor with k = 1.

(Q11)
Normalizing an unbound distribution like chi-squared using min-max is a bad idea because min-max scaling is sensitive to outliers. If a chi-squared distributed feature has extreme values, they can distort the scale for the rest of the data. This can lead to loss of information about patterns in the data. We get different results each time depending on how extreme the outliers are in the specific sample that we have each time, which will result in a high variance

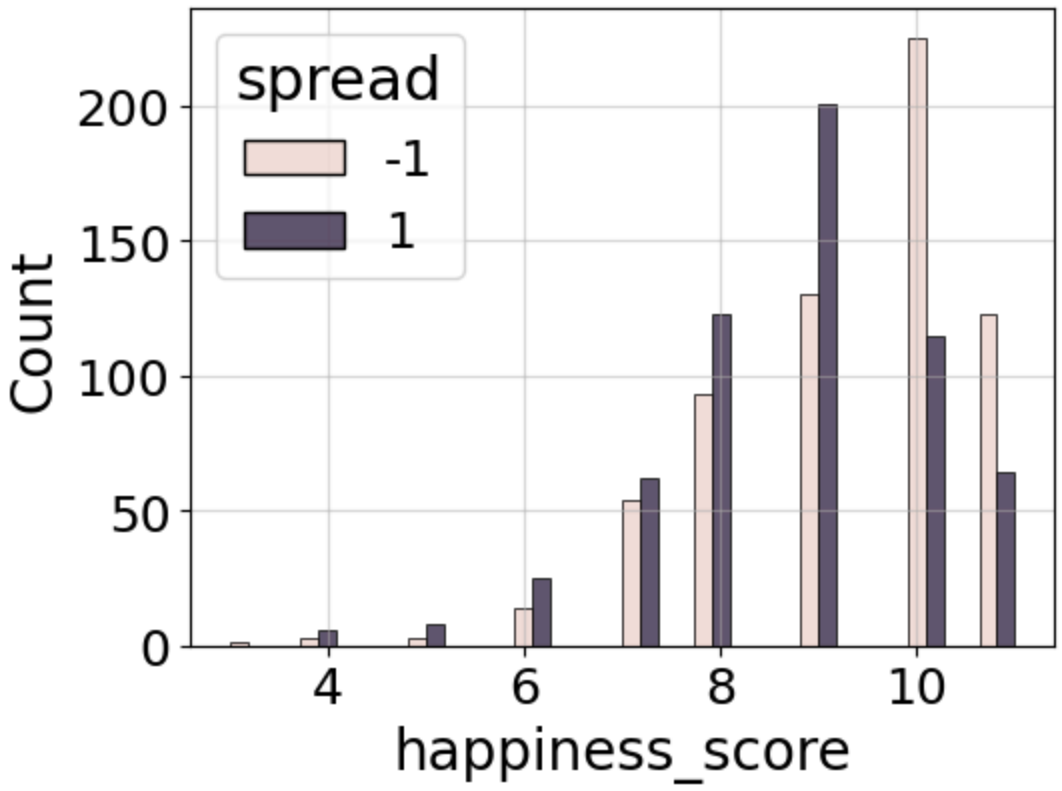
Part 3: Data Exploration

(Q12)
We will need 1 feature containing 8 boolean values. Because there are 8 possible values for blood_type: | {+, -} X {A, B, AB, O} | = 8.

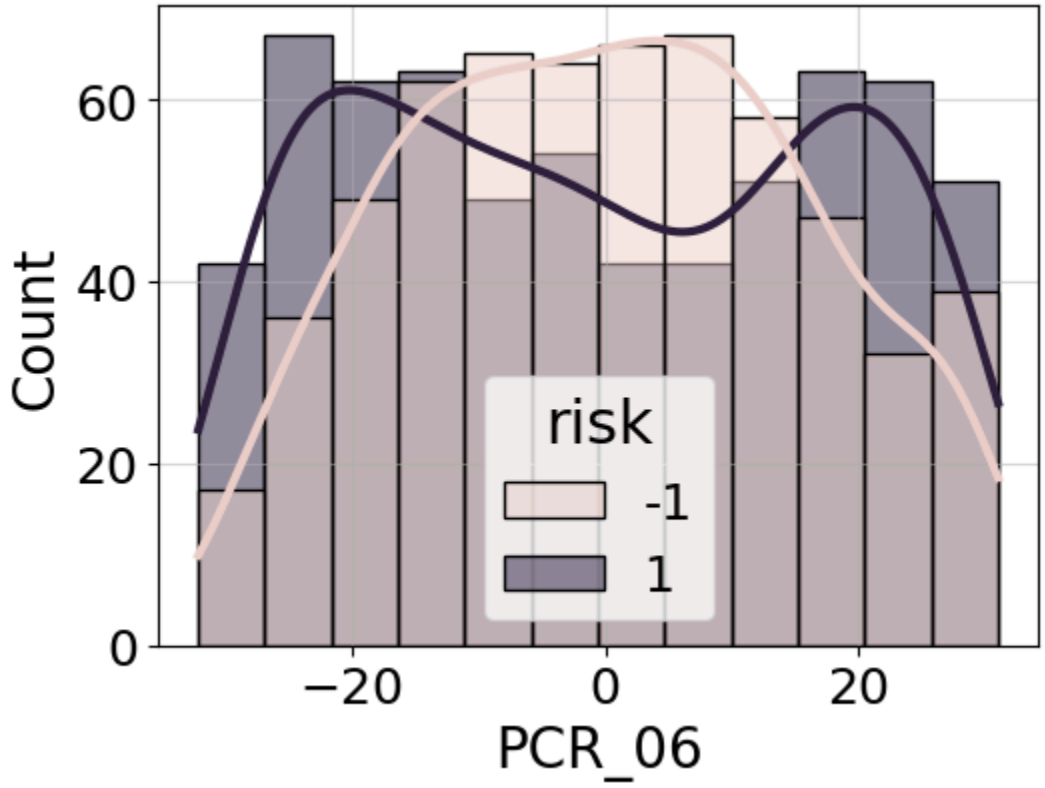
(Q13)
Yes we can. We can replace the values of the “symptoms” feature with a numeric value that indicates how many different symptoms a patient is experiencing, which may hint at how severe the symptoms are.
To craft this new feature, all patients with nan get 0, for the rest of the patients, we count the occurrences of ‘;’ + 1.

(Q14+Q15)
The most informative features for predicting the *spread and risk* are the features with a plot with clear dominance of one value (either 1 or -1) of the spread/risk in different regions of values of the feature which indicates that this feature can be used to predict the value of the spread/risk.

The most informative feature for predicting the *spread* target variable is *happiness_score.*

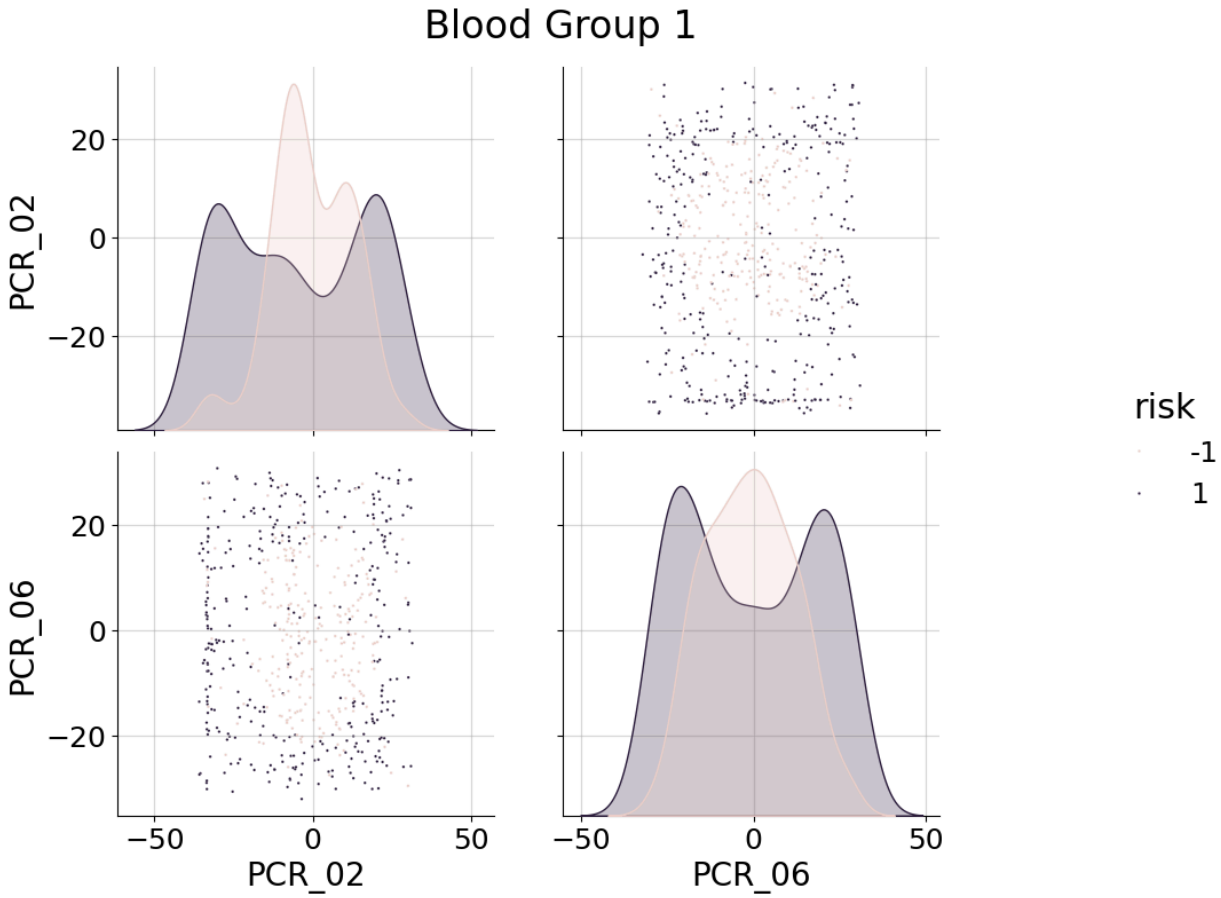


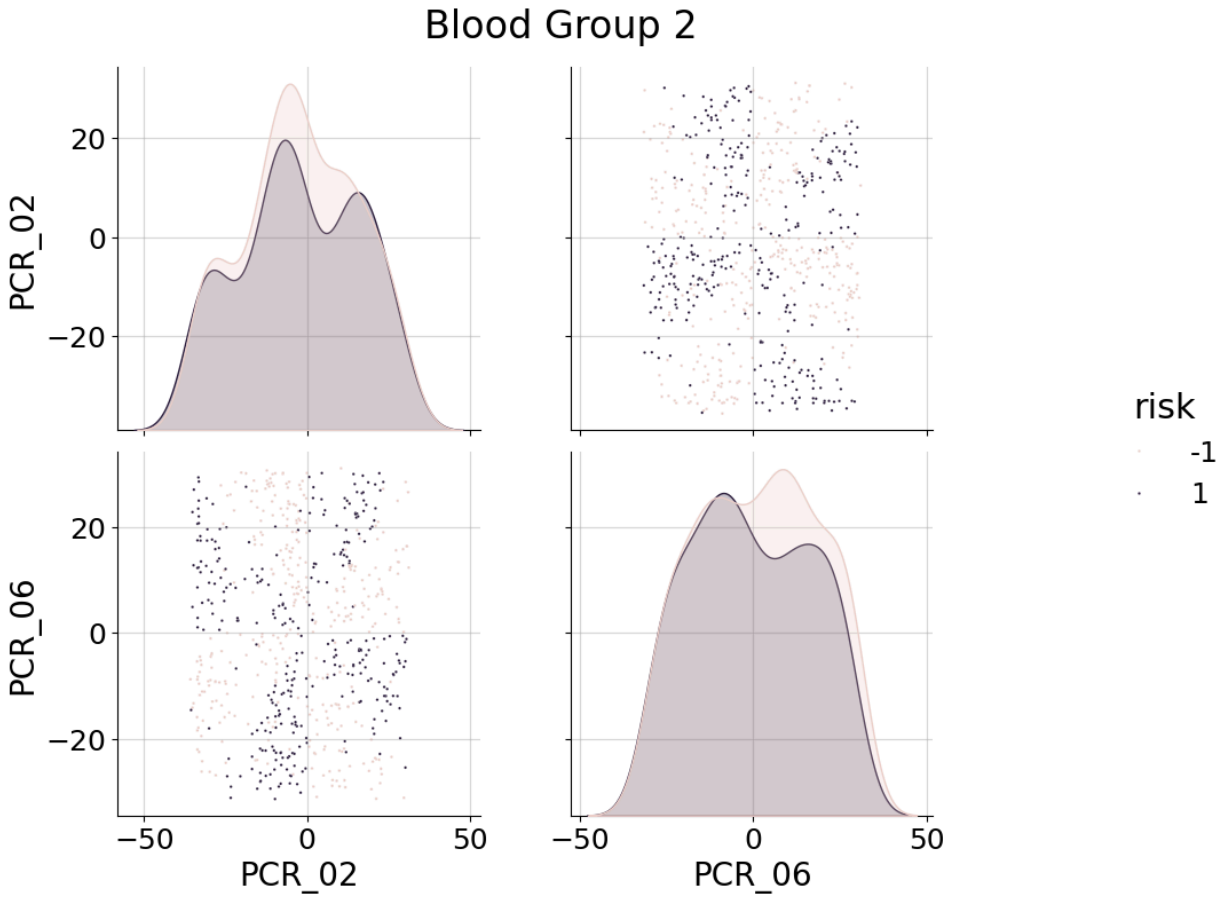
The most informative feature for predicting the *risk* target variable is *PCR_06.*



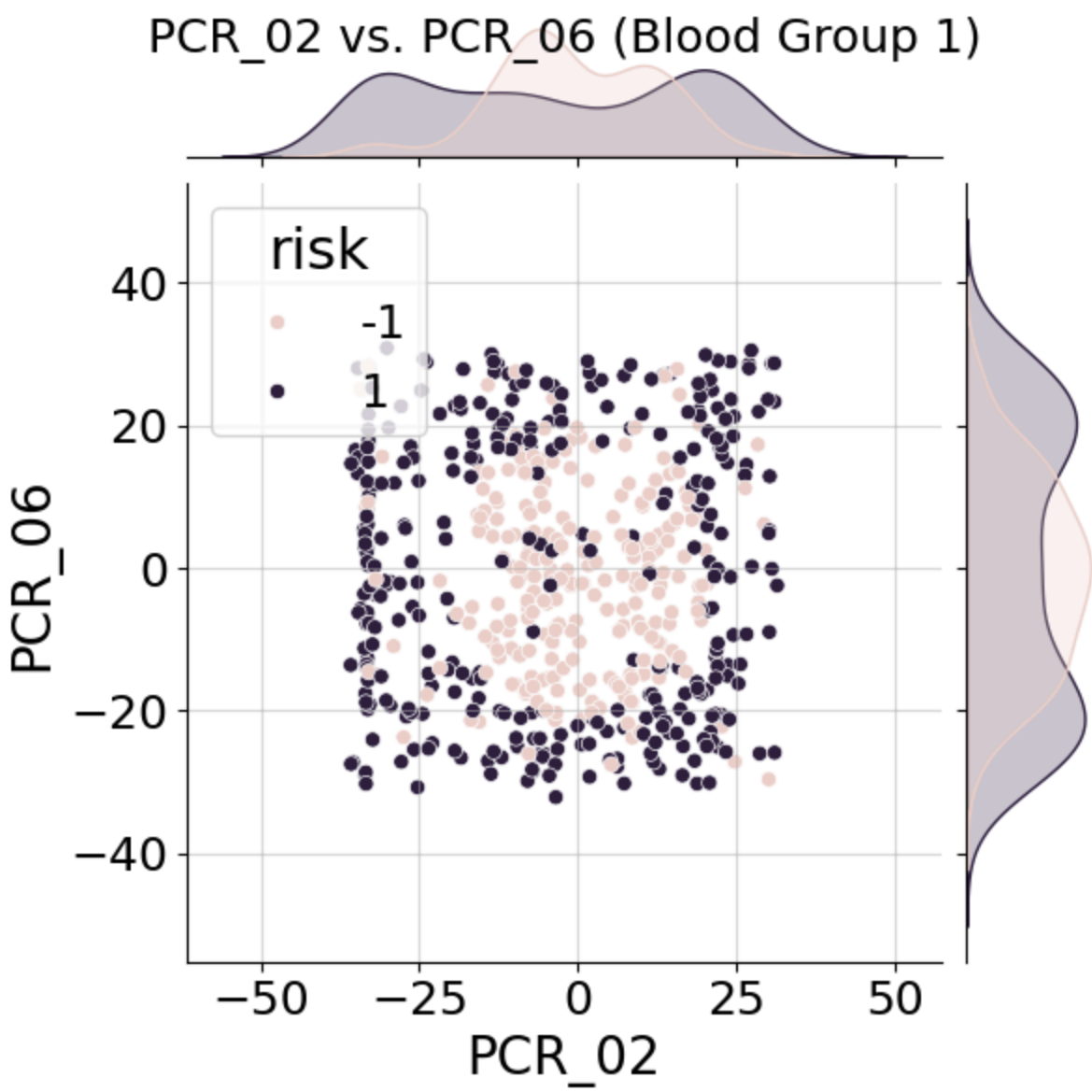
(Q16)

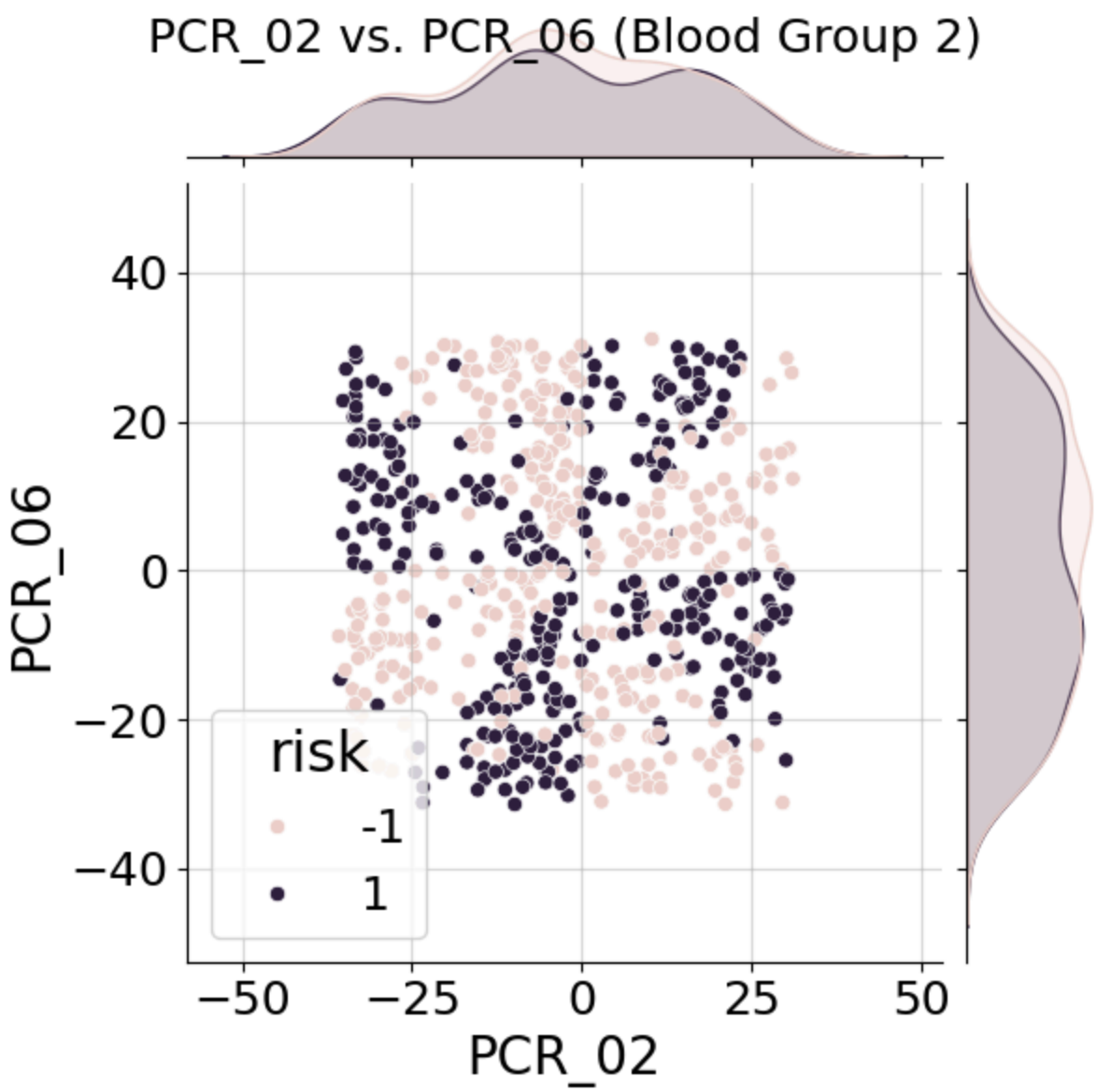
PCR_02 and PCR_06, we can see clear separability in the plots for both boold groups

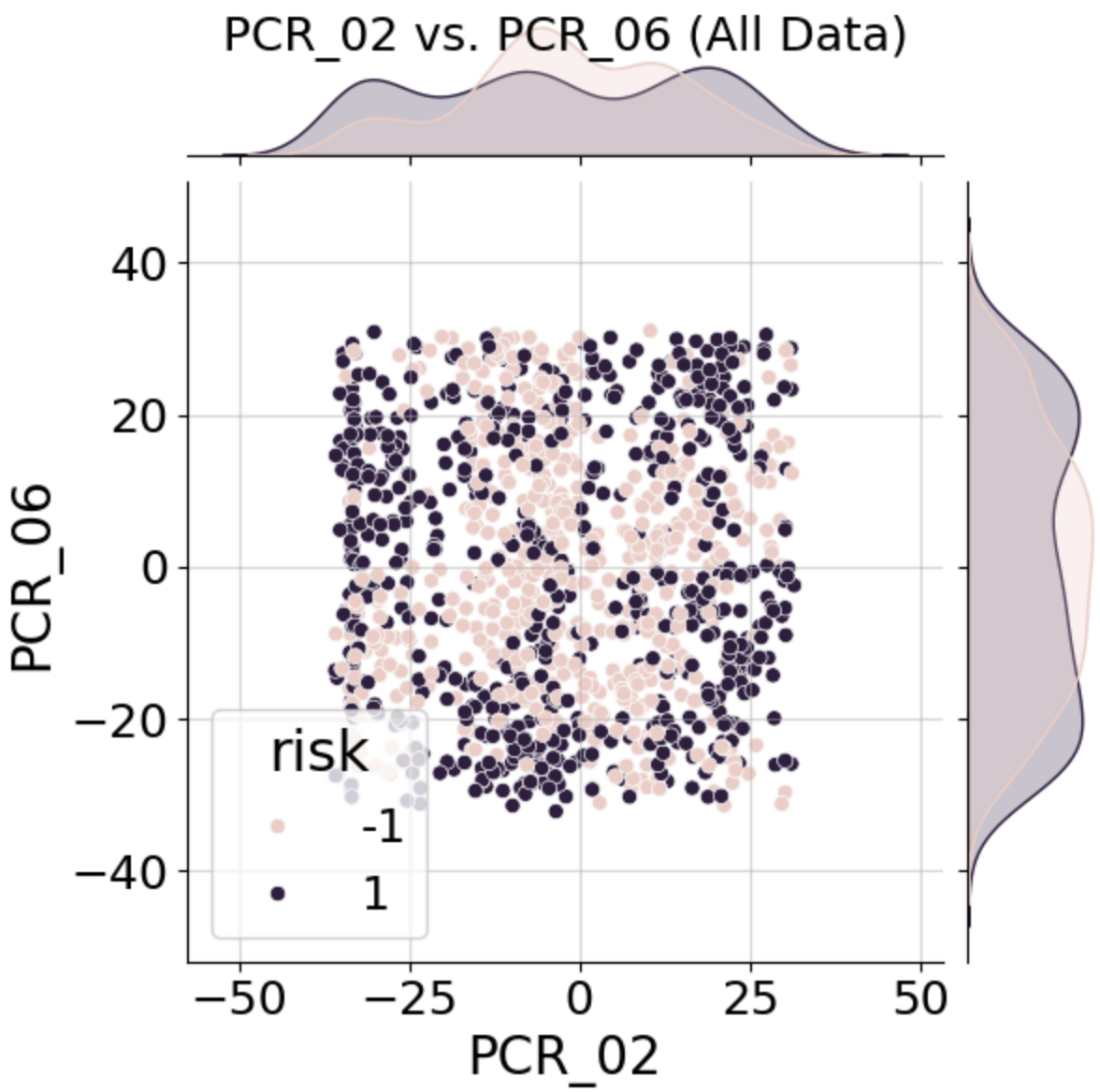




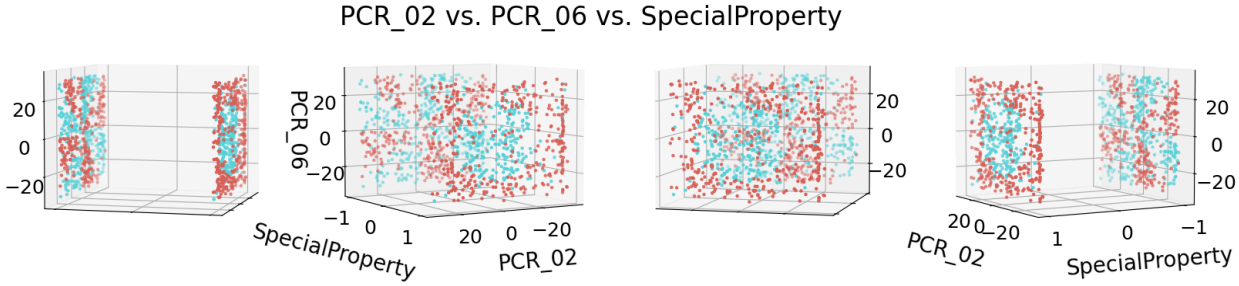
(Q17)


PCR_02 vs. PCR_06 (Blood Group 1)
risk
-1
1
40
20
0
−20
−40
PCR_06
−50
−25
0
25
50
PCR_02


PCR_02 vs. PCR_06 (Blood Group 2)
PCR_06
PCR_02
risk
-1
1
40
20
0
−20
−40
−50
−25
0
25
50


PCR_02 vs. PCR_06 (All Data)
PCR_06
PCR_02
risk
-1
1
40
20
0
−20
−40
−50
−25
0
25
50

(Q18)


PCR_02 vs. PCR_06 vs. SpecialProperty
PCR_06
SpecialProperty
PCR_02
PCR_02
SpecialProperty

(Q19)

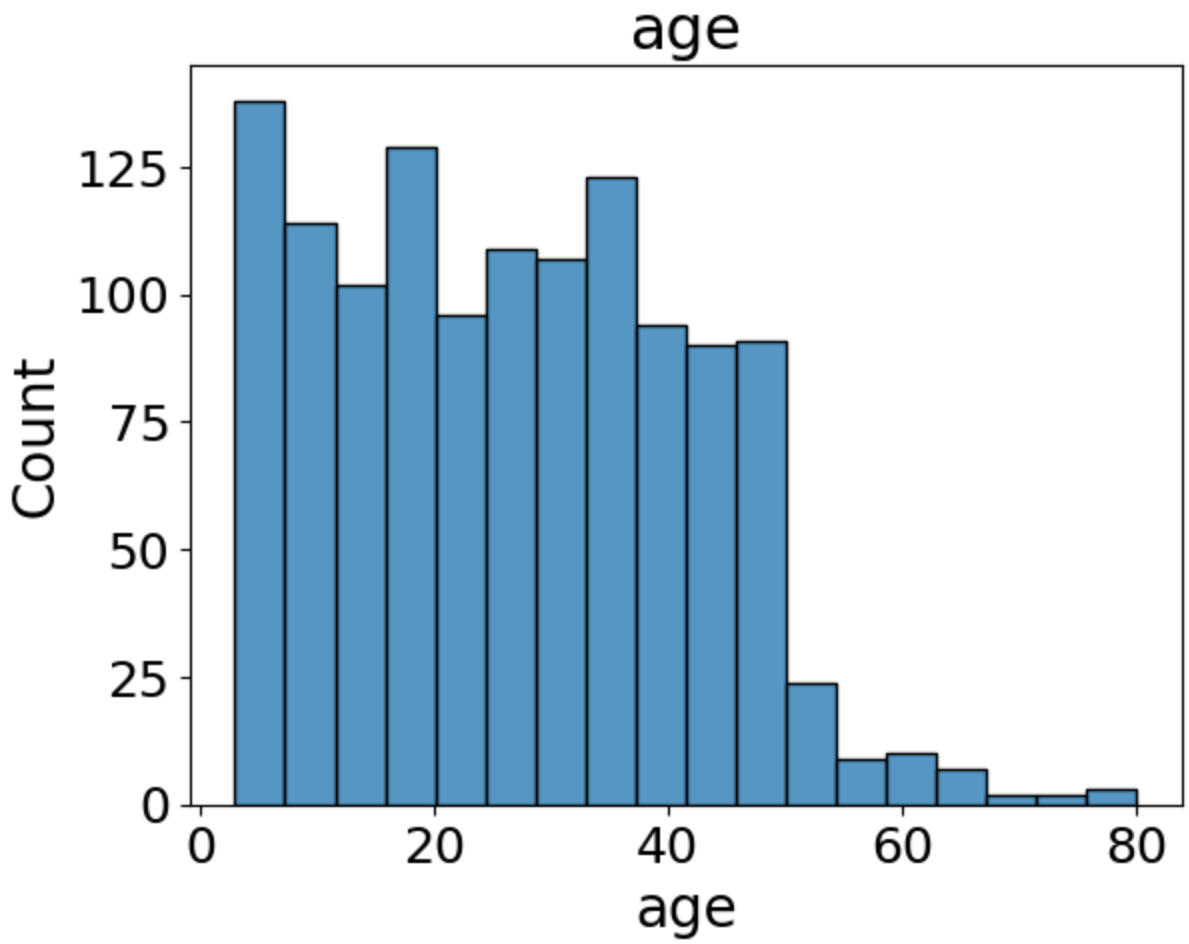
A decision tree of depth 3 won't be able to predict the risk well since we need one depth for blood groups and many regions in the PCR_02 vs PCR_06 plot that we can't predict accurately with only 2 binary splits.

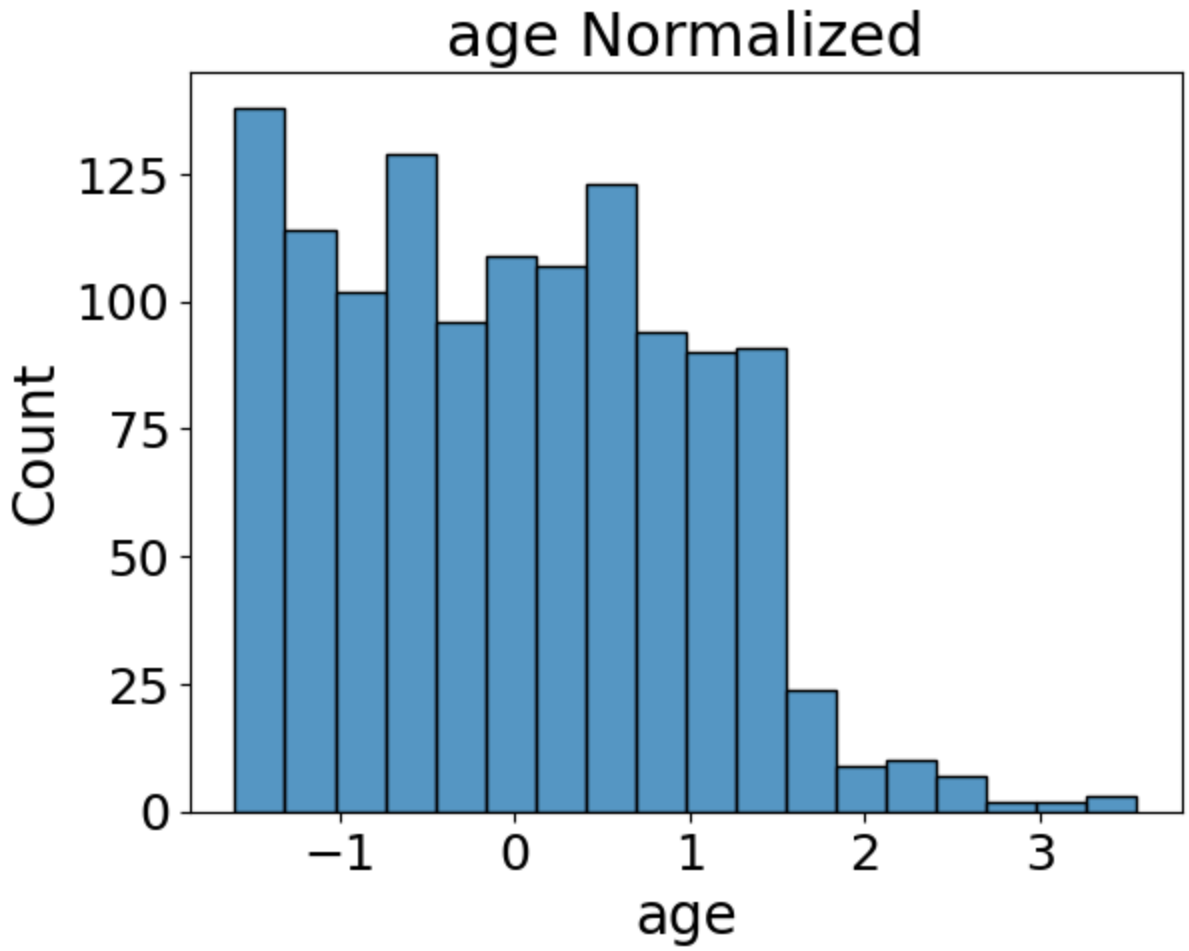
(Q20)
We can accurately predict the risk since with 30 depth we can accurately split the regions in the PCR_02 vs PCR_06 plot.

(Q21)
A 1-NN model will be accurate only if we scale the SpecialProperty value to prevent points from one blood group to be nearest neighbors of points from the other blood group, that way we can accurately predict the risk value as was seen in previous questions.

Part 4: More Data Normalization

(Q22)


age
Count
125
100
75
50
25
0
0
20
40
60
80
age



(Q23)
In the case of a decision tree, be it of max-depth=3 or 30, its decision boundaries get normalized along with the features, therefore do not actually change relative to the data. Meaning, normalization does not affect the answers in Q19 and Q20.

The 1-NN model can improve but not a lot, because the scales of the featuresPCR_02 and PCR_06 are mostly in the same range (+-40 and +- 50). So normalizing the data won't affect the distances between the points too much, and therefore the Knn model doesn't get affected too much either.

(Q24)
***Reasoning for normalization method after table**

Feature Name	keep	new	Normalization method	explanation

patient_id	v	x	-	-
age	v	x	Standard	-
sex	v	x	-	Boolean values were converted to -1 , 1.
weight	v	x	Standard	-
blood_type	x	x	-	Dropped in favor of SpecialProperty.
SpecialProperty	v	v	-	Bunched all blood types of O+ and B+ into one numerical category represented by 1. And the rest of the blood types into a numerical category represented by -1. Explained in Q12.
current_location	x	x	-	Was split to 2 features: x_location and y_location.
num_of_siblings	v	x	Standard	-
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	v	Standard	We changed the values from strings separated by ‘;’ to integer counting the number of symptoms a patient has. As explained in Q13.
pcr_date	v	x	MinMax	
x_location	v	v	Standard	Split the current_location

				feature and inserted its first value into this one.
y_location	v	v	Standard	Similar to previous feature, inserted second value of current_location into this one.
PCR_01, PCR_2, PCR_03, PCR_5, PCR_06	v	x	MinMax	-
PCR_04, PCR_07, PCR_8, PCR_09, PCR_10	v	x	Standard	-

We decided to use StandardScaler on all features that seem to exhibit a mountain-like plot (like PCR_04), since the edges of the mountains represent outlier data which we would like to neglect in favor of the majority data in the bulk of the mountain. And StandardScaler is the better normalization for capturing the majority of data in a desired range like -1 to 1.

We decided to use MinMaxScaler on the remaining features that appear to be more uniformly distributed (like PCR_01), and we can simply normalize the data to a desired range whilst maintaining the data distribution without disruptions from potential outliers.