ML Major 1

Submitters:

Husain Azaqy: 213382104
Galor Lazar: 325028876

(Q1) Load the dataset into a Pandas DataFrame.

Answer (in your report)**:** how many rows and columns are in the dataset?

25 columns

1250 rows

(Q2) Print the value_counts of the conversations_per_day feature (see Tutorial 01). Copy the obtained output to your report. Describe in one short sentence what you think this feature refers to in the real world.

This feature's type is "ordinal". Explain briefly why.

```
conversations_per_day
3     218
2     204
5     179
4     168
1     108
6     107
7      94
8      54
9      42
10     29
11     16
13      8
12      7
14      6
16      5
15      3
17      1
29      1
Name: count, dtype: int64
```

We think this feature refers to how many times a person spoke to other people that they can infect with the virus. It is an ordinal variable since the count of conversations is a natural number with natural order as a counting variable.

(Q3) In your report, write a table describing each feature. The columns must be:

a. Feature name: the name of the feature as it is written in the dataset.
b. Description: a short sentence with your understanding of the feature's meaning in the real world.
c. Type: Continuous, Categorical, Ordinal, or Other.
Don't overthink this (especially the "ordinal" type), some variable may be suitable for two types.

Note: do not include the target columns ("spread" and "risk").

Name	Description	Type
patient_id	Unique identifier for each patient	Ordinal
age	Patient age in years	Ordinal
sex	Patient gender	Categorical
weight	Patient weight in kilograms	Continuous
blood_type	Patient blood group	Categorical
current_location	Patient present location	Continuous (coordinates)
num_of_siblings	Number of patient's siblings	Ordinal/Categorical
happiness_score	Patient reported happiness level	Ordinal
household_income	Annual/Monthly income of patient's household	Continuous
conversations_per_day	Daily conversations count for patient	Ordinal
sugar_levels	Patient blood sugar levels	Ordinal
sport_activity	Frequency of the patient's sports participation	Ordinal
pcr_date	Date of PCR test	Ordinal
PCR_01	PCR test result 1	Continuous
PCR_02	PCR test result 2	Continuous
PCR_03	PCR test result 3	Continuous
PCR_04	PCR test result 4	Continuous
PCR_05	PCR test result 5	Continuous
PCR_06	PCR test result 6	Continuous
PCR_07	PCR test result 7	Continuous
PCR_08	PCR test result 8	Continuous
PCR_09	PCR test result 9	Continuous
PCR_10	PCR test result 10	Continuous

(Q4) Split the data randomly into a training set (80% of the data) and a test set (20% of the data). As the random_state, use the sum of the last two digits of each of your IDs[1] (two or three IDs).

The random state will ensure that you get the same split every time. Answer: Why is it important that we use the exact same split for all our analyses?

- Since the default split is random, we want the same random selection between multiple analyses to be able to compare them reliably and easily.

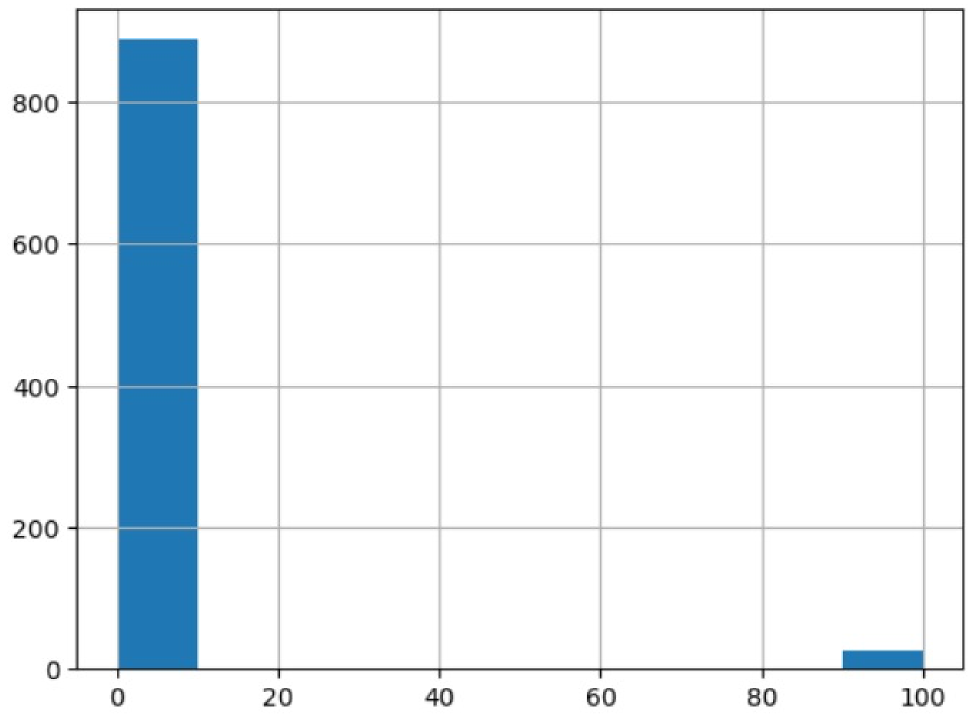
(Q5) For **both the training set and test set**, report which fields have missing values and how many missing values there are. You can use Panda's function isnull().

- The only field that has missing values is " household_income" with 109 missing values.
- 86 of them are in the training set.
- 23 of them are in the test set.

(Q6) Plot a histogram (see Tutorial 01) for each field where you found missing values in **(Q5)**. Add these plots to your report. Answer: Can you recognize outliers? **Reminder:** Create plots using only the training set.

```
train_virus_df["household_income"].hist()
```

```
<Axes: >
```



There are outliers as we can see in the far-right corner of the plot.

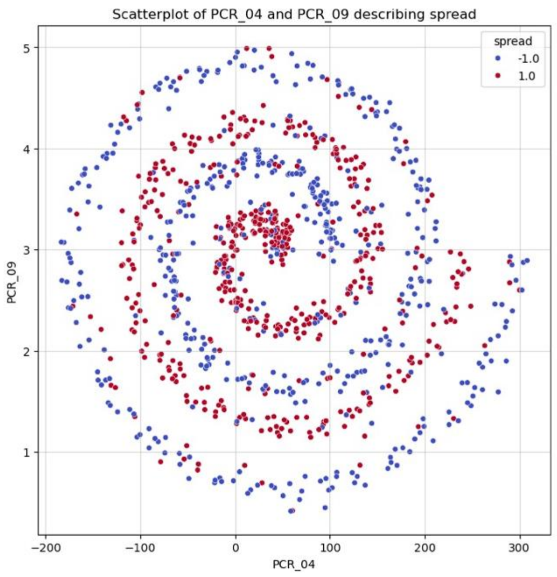
(Q7) For each field where you found missing values, calculate the median and the mean in the training set, **and report it**.

If there is a significant difference between the mean and median values, explain the reason. Which filling method do you prefer to use in our case, and why?

- The mean household income 3.535886214442013
- The median household income 0.7
- We found that 888 of the 914 values in the training set are under the mean household-income. It means that the mean value doesn't represent a good average household-income value. That is in corollary to the extremely far outliers in the training data plot.
- As a result we choose to fill the NaN's/nulls with the median, that is closer to most of the current elements.

(Q8) Answer briefly: Based on the plots you created on **(Task B)**, what pair of features is useful for predicting the `spread`?

Attach the `seaborn.pairplot` of only this pair of features to the report. Make sure your plots are readable and clear, and that they have proper titles, grid lines, axis labels, etc. Any missing requirement will lead to a points deduction!



We chose the plot of {PCR_04,PCR_09}, because it is the most sparable one. We can separate it by spiral function (function that is similar to "Archimedean spiral" function).

(Q9) What is the time complexity of the prediction function you wrote, applied on a single test datapoint, in terms of the number of neighbors k, the number of training datapoints m and the data dimension d? Explain. It is okay to "estimate" the complexity of python library functions. For instance, if you use `np.argsort` on n elements, then its complexity should be $O(n \log \log n)$. Use your reason and CS knowledge.

We will break the complexity to parts:

- Computing distances:
  - Computing for m training data points the distance from a fixed point is O(d) operations per point. Totaling O(m*d)

- The function np.argpartition:
  - We estimate it has time complexity O(m) on average case, and O(m^2) in the worst case.We assume it used the quick select algorithm for finding the k-th element, then going over the rest of the elements it added only the elements less than the k-th element to the left part of the partition. Done in O(m) worst case.
  - Total O(m) amortized.

  Note:

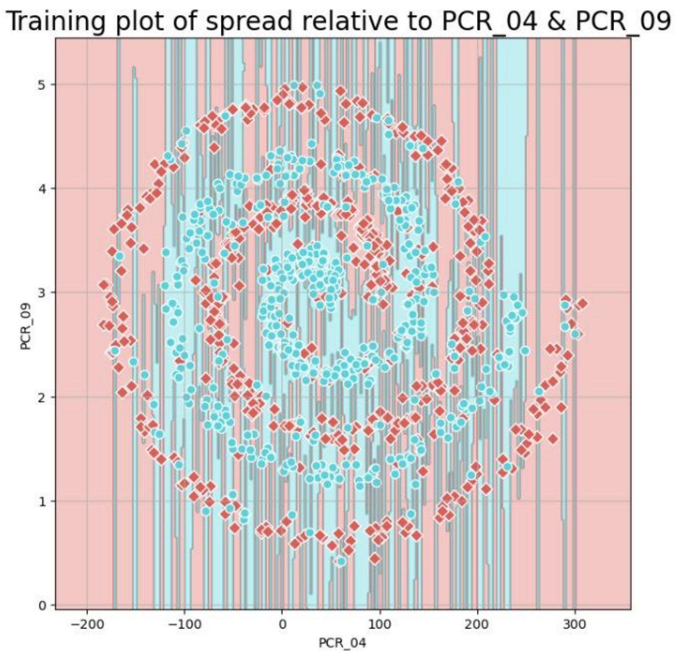
  - Using merge sort or any sort with O(mlogm) complexity , the task can be done in O(mlogm) worst case, instead of O(m^2) worst case with quick select.
- Taking the smallest k elements: O(k)
- Summing all elements and taking the sign: O(k)

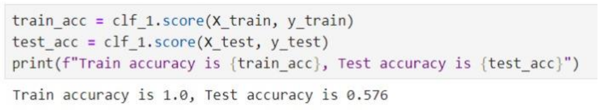
Total:

- On average : (quick select) O(m*d+m+k), in other words: O(m*d) since k<=m.
- Worst case: (Sorting)) O(mlogm + m*d +k), since k<=m, we really get : O(mlogm+m*d)

(Q10) Attach the figure to your report. Specify the model's training and test accuracies.

(The plot should exhibit a bizarre behavior which we will discuss next.)





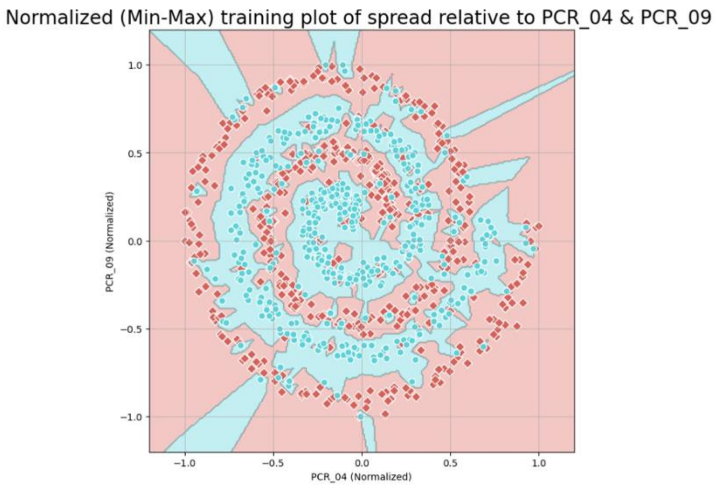
```
train_acc = clf_1.score(X_train, y_train)
test_acc = clf_1.score(X_test, y_test)
print(f"Train accuracy is {train_acc}, Test accuracy is {test_acc}")
```

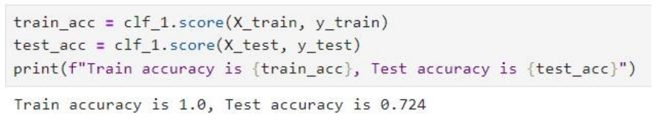
```
Train accuracy is 1.0, Test accuracy is 0.576
```

The model reaches a 100% accuracy rate on the training set, which is expected since it was trained on this dataset. The model identifies the closest datapoint in the training data for each input, It finds the point itself and therefore perfect accuracy is reached.
The model got an accuracy of 57.6% on the test set. Unlike the training set, the algorithm didn't know this data before, requiring it to make predictions rather than relying on memorized data (like in the training set), hence the lower accuracy.

(Q11) Use min-max scaling (between $[-1,1]$) to normalize the two features in the temporary `DataFrame` you created before, and train a new kNN model ($k = 1$) on the <u>normalized</u> dataset.

Compute the new training and test accuracies and draw the decision regions of the model. Attach the results to your report and compare them to those from **(Q10)** for the same $k = 1$ model on the raw data. Use these results to explain why normalization is important for nearest neighbor models.





```
train_acc = clf_1.score(X_train, y_train)
test_acc = clf_1.score(X_test, y_test)
print(f"Train accuracy is {train_acc}, Test accuracy is {test_acc}")
```

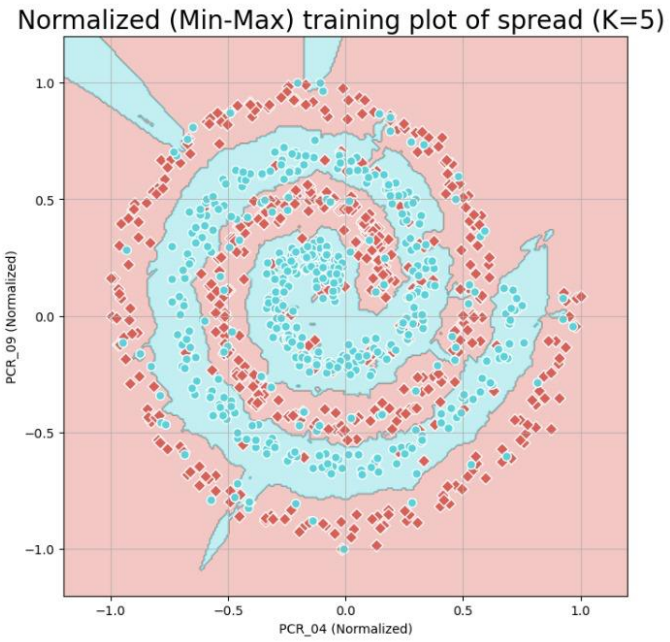
```
Train accuracy is 1.0, Test accuracy is 0.724
```

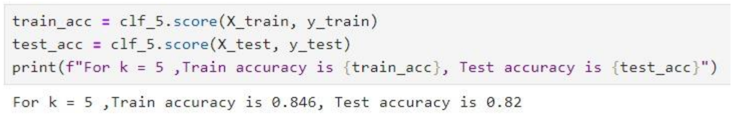
From the same reasons of Q10 we got 100% accuracy on the training set.
The model got an accuracy of 72.4% on the test set. From the same reasons in Q10 we got lower accuracy on the test set.

Normalization using the min max method results in equal weighting of features, meaning it ensures that each feature contributes equally to the distance computation of the model. When features are on different scales (Like PCR_04, PCR_09, the mean of PCR_04 is 14.2 times larger than the mean of PCR_09), those with larger ranges skew the distance calculations, overpowering the effect of features with smaller scales. Normalizing the features makes it so that each feature is given equal consideration.

The improved results are because normalized data makes sure that the optimization process moves more smoothly through the feature space, avoiding issues where features with larger scales unfairly affect the direction and magnitude of the predictions.

(Q12) Using the normalized dataset, train another kNN model with $k = 5$. Compute the training and test accuracy and draw the decision regions of this model.
Attach the results to your report and compare them to those from **(Q11)**. Use these results to briefly explain the effect of k on the decision regions.





```
train_acc = clf_5.score(X_train, y_train)
test_acc = clf_5.score(X_test, y_test)
print(f"For k = 5 ,Train accuracy is {train_acc}, Test accuracy is {test_acc}")
```

```
For k = 5 ,Train accuracy is 0.846, Test accuracy is 0.82
```

Unlike the previous questions, now we got 84.6% accuracy on the training set, because this time we work on 5-NN model.
The model got an accuracy of 82% on the test set – better than previous models.

We got decrease in training set accuracy because the 5-NN model averages the labels of the five nearest neighbors, leading to less overfitting. The 5-NN model makes predictions based on a wider context from multiple neighboring datapoints rather than a single one.

So the 1-NN model perfectly memorizes the training data but doesn't generalize well to new data. The 5-NN model, is less accurate on the training set but predicts better to the test set, resulting in improved predictions accuracy.

(Q13) This question is general and does not deal with the given dataset. Assume a dataset with two features, one randomly sampled (i.i.d.) from a uniform continuous distribution on the range $[2,5]$ and the other randomly sampled (i.i.d.) from a chi-squared distribution $\chi^2(k=2)$ (see in Wikipedia).

(The labels are determined by some unknown function of these two features.)

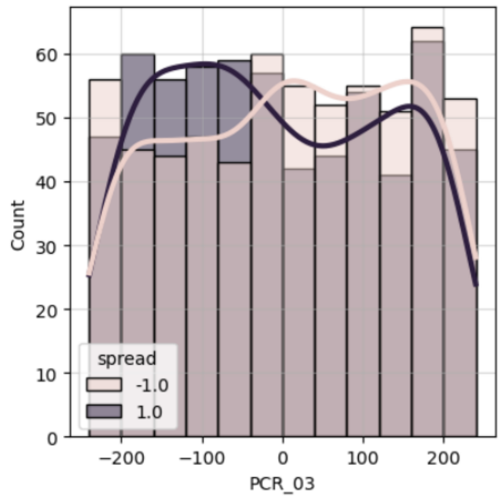
Why is normalizing both features using min-max scaling to $[-1,1]$ a bad idea? Explain in detail.

Normalizing both features using min-max scaling to [-1,1] is problematic because the two distributions are fundamentally different in their range and distribution shape. The feature sampled from the U[2,5] distribution has a “small” and known range, while the feature from the chi-squared distribution has lower bound at zero and a potentially long “tail”. Those differences means that min-max scaling will compress the uniform feature, but it will distort the chi-squared feature's distribution, by taking most values into a small range and overemphasizing outliers. So the normalized features won’t be on a comparable scale, which can confuse the model and hurt its performance.

(Q14) According to the univariate analysis, name one feature that seems informative for predicting the `spread` target variable (other than the 2 features from **Q8**).

Attach the appropriate univariate plot and briefly explain (2-3 sentences) why this plot makes you think that feature is informative.

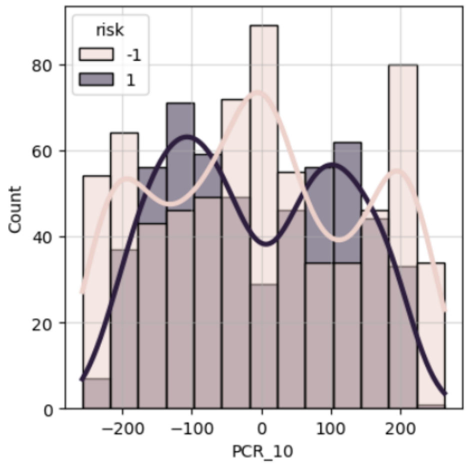
The feature we choose besides PCR_04 PCR_09, is PCR_03. Since it demonstrates better distinction between spread values of 1 and -1 than PCR_01 and PCR_02 for example.



(Q15) According to the univariate analysis, name one feature that seems informative for predicting the `risk` target variable (other than the blood groups).

Attach the appropriate univariate plot and briefly explain (2-3 sentences) why this plot makes you think that feature is informative.

We choose the feature PCR_10, (can choose PCR_03 again) since again, for the most part it distinguishes better between risk values of 1 and -1 than other features.



(Q16) Split the (training) data based on the binary `SpecialProperty` feature created in **(Task E)**. For each split, perform a bivariate analysis for the PCR features in the set**,** in relation to the `risk`. This means you should produce two "matrices" of plots, one for each blood group. Each matrix should contain 4x4 subplots, representing all possible pairs of PCR features in the set. Do not include these plots in your report.

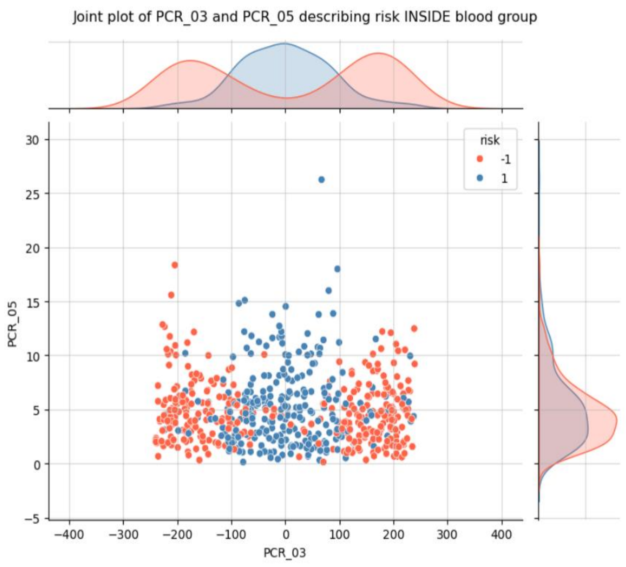
According to those plots, choose a pair of `PCR` features that could be helpful for predicting the `risk` (with the partition according to the `SpecialProperty`). What `PCR` features did you choose? And why?

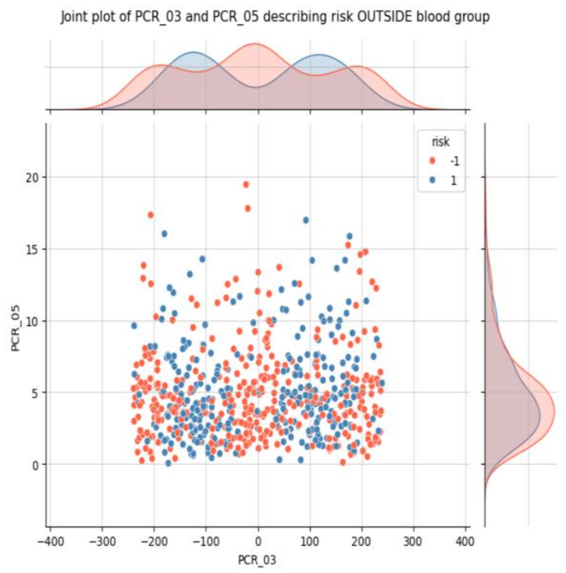
The pair of features we will choose are PCR_03, PCR_05 . Sice they separate the risk target well regardless of the SpecialProperty value.

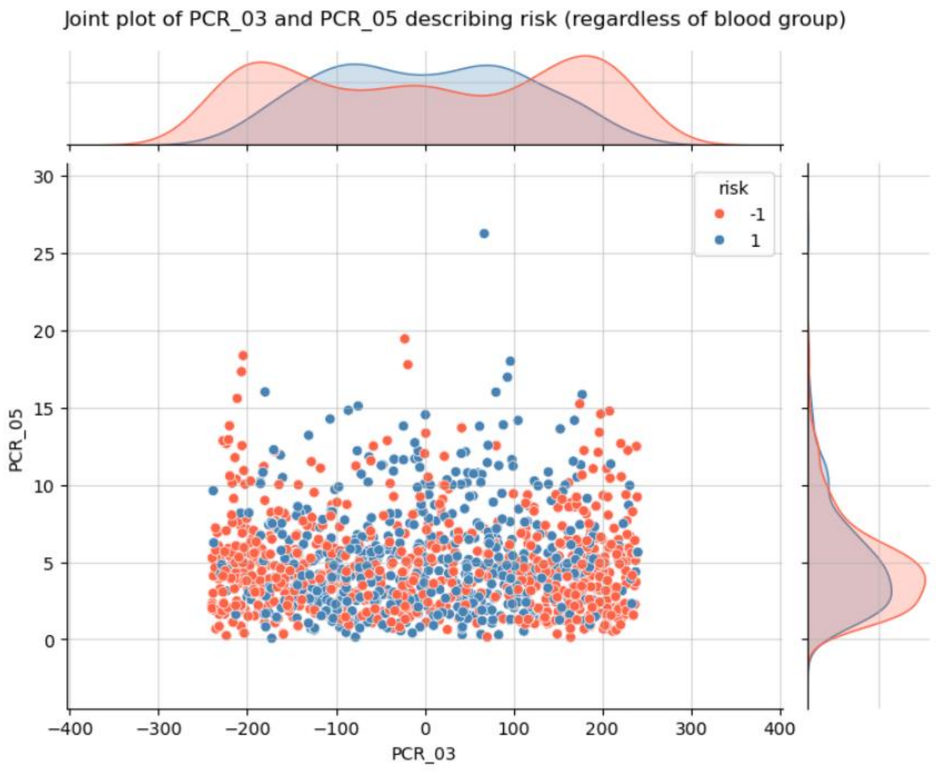
Other pairs seem to separate well if the SpecialProperty holds but not so well if it doesn't. or vice versa.

(Q17) For the pair of `PCR` features you chose in **(Q16)**, create three `jointplot`s (see Tutorial 01), all conditioned on the `risk` variable. The first `jointplot` should include only the data in the first blood group you created in **(Task E)**, `{O+, B+}`. The second `jointplot` should include only the data in the other blood group. The third `jointplot` should be for the full data, without partitioning to blood groups. Attach the 3 resulting plots to your report. Remember to have grids, titles, and axis-labels.

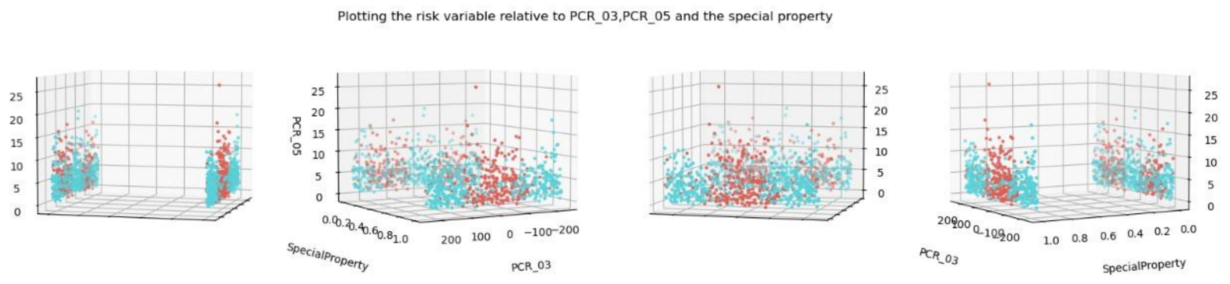
The plots:







(Q18) Use the provided function `plot3d` to plot the pair of `PCR` features you chose (axes X and Z) and the `SpecialProperty` feature (axis Y), colored by the `risk` label. Make sure that the plot is clear & readable and that it has a proper title. Attach the plot to your report.



(Q19) How well will a decision tree of `max-depth=3` be able to fit the <u>training</u> data? Explain briefly.

A decision tree with max-depth = 3 will not fit the training data well. It can make up to 8 splits (3 questions in the decision tree), allowing some partitioning of the data based on the three features. As a result, it might not fully fit the data, leading to underfitting the risk variable.

(Q20) How well will a decision tree of `max-depth=30` be able to fit the <u>training</u> data? Explain briefly.

A decision with max-depth = 30 will fit the training data very well. With such depth, the model can make many splits (many questions in the decision tree) and create very detailed patterns in the data. This allows us to fit a datapoint in the training data almost perfectly by asking 30 questions at most on it and predict it’s risk . So it will have a very low training error.

(Q21) How well will a `1-NN` model be able to fit the training data? Note that in this question, a point in the training set is not considered its own neighbor (i.e., when making a prediction for a training data point, the model won't use the same point for prediction, but only the nearest point in the remaining training set).

Hint: consider the scale of the features in your answer.

Assuming no two points have the same features but different labels, in the normal case we will get 100% accuracy. In contrast, if we don't consider the point itself as the closest point to itself, we could get a non-perfect score on the training data.
For example, if the training data contains two points with different features and different labels, each point would be labeled worng and we will get 0% accuracy on the training data.

{ Task F scaling decisions: }

MinMaxScaler	StandardScaler
PCR_03	PCR_01
PCR_10	PCR_02
	PCR_04
	PCR_05
	PCR_06
	PCR_07
	PCR_08
	PCR_09

(Q22) What will be the effects of data normalization on your answers in **(Q19)**, **(Q20)**, **(Q21)**?

Q19+Q20:
For both a decision tree with a max-depth = 3 and with max-depth = 30, data normalization will have almost no effect. Decision trees make their splits based on feature values rather than distances, making them very insensitive to the actual scale of the features. While normalization might have a big influence on the choice of split points, the overall structure and performance of the trees will remain mostly unchanged. (In a sense, the decision boundaries made by a tree is perpendicular to a feature axis)

Q21:
The normalization will have a very large effects on the 1-NN model and it may do a significant improvement to performance of predictions. By scaling the features to a similar range, the model will use all the PCR features more equally when it will calculate the distances. This correction should help the model to better identify the nearest neighbors based on a balanced consideration of all the features. It may lead to more accurate predictions and better splitting and handling areas where risk values change.
However, normalization could also harm the model if it removes important feature distinctions or if the original feature’s scales were optimal for capturing data patterns. Also if the features had meaningful relationships on their original scales, the normalization might harm these, leading to less accurate predictions.
So normalization is most likely to enhance performance by addressing the scale diversity, but it may also unintentionally reduce the importance of certain aspects of the data.

(Q23) Write a table summarizing the data preparation process you created.

The columns of the table must be:

a. **Feature name**: the name of the feature as written in the dataset.
   Names of new features should be meaningful!
b. **Keep**: "V" if the feature is kept, "X" otherwise (e.g., `blood_type` is removed).
c. **New**: "V" if the feature was handcrafted using other feature(s), "X" otherwise.
d. **Normalization method,** if used.

Feature Name	Keep	New	Normalization method
patient_id	V	X	-
age	V	X	-
sex	V	X	-
weight	V	X	-
blood_type	X	X	-
current_location	V	X	-
num_of_siblings	V	X	-
happiness_score	V	X	-
household_income	V	X	-
conversations_per_day	V	X	-
sugar_levels	V	X	-
sport_activity	V	X	-
pcr_date	V	X	-
PCR_01	V	X	Standard Scaling
PCR_02	V	X	Standard Scaling
PCR_03	V	X	MinMax scaling
PCR_04	V	X	Standard Scaling
PCR_05	V	X	Standard Scaling
PCR_06	V	X	Standard Scaling
PCR_07	V	X	Standard Scaling
PCR_08	V	X	Standard Scaling
PCR_09	V	X	Standard Scaling
PCR_10	V	X	MinMax scaling
Special Property	Not applicable	V	-