Homework

Course: Introduction to Machine Learning

Assignment: Major HW1

Submitters: Ofer Nissim – 312367576

Lahav Fridlander – 209403781

Date: 30/11/22

Part 1: Data Loading and First Look

(Q1) Load the dataset into a Pandas `DataFrame`.

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

The dataset has 1250 rows and 26 columns.

(Q2) Print the `value_counts` of the `num_of_siblings` feature (see Tutorial 01). Copy the obtained output to your report. Moreover, 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.

Remember to clearly write the number of the question next to your answer.

The obtained output:

```
[10] print( dataset["num_of_siblings"].value_counts() )

     1    399
     2    317
     0    271
     3    161
     4     62
     5     31
     6      6
     7      2
     8      1
     Name: num_of_siblings, dtype: int64
```

We think this feature refers to the number of siblings a patient has.

This feature is ordinal because the number of siblings a person can have is a natural number, which also has a natural order (starting from 0).

(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 to include the target columns ("`spread`" and "`risk`").

The table:

Feature name	Description	Type
patient_id	Patient's ID	Ordinal
age	Patient's age	Ordinal
sex	Patient's sex	Categorical
weight	Patient's weight	Continuous
blood_type	Patient's blood type	Categorical
current_location	Patient's current location – represented as coordinates	Other
num_of_siblings	Patient's number of siblings	Ordinal
happiness_score	Patient's happiness level, as rated by the patient	Ordinal
household_income	Patient's household income	Continuous
conversations_per_day	Number of conversations the patients has per day	Ordinal
sugar_levels	Patient's sugar levels	Ordinal
sport_activity	Number of sport activities made by the patient per week	Ordinal
symptoms	Patient's symptoms, as described by them	Other
pcr_date	Date of the latest PCR test	Ordinal
PCR_i (i=1,…,10)	Results of the i-th PCR test	Continuous

Partitioning the data

(Q4) Split the data into a training set (80%) and a test set (20%). As the `random_state`, use the sum of the last two digits of your i.d and your partner's i.d[1].
The random state will ensure that you get the same split every time. Why is it important that we use the exact same split for all our analyses?

Last two digits of our IDs are 76 and 81, hence the sum for the random_state is 157.

When using the exact same split for our analyses, we can ensure that the subsets of both the training and the test will stay the same (i.e., no data "migration" between the sets) each time we access them. Having that, it will result in a "neutral" test, where the learned model isn't affected by the test data.

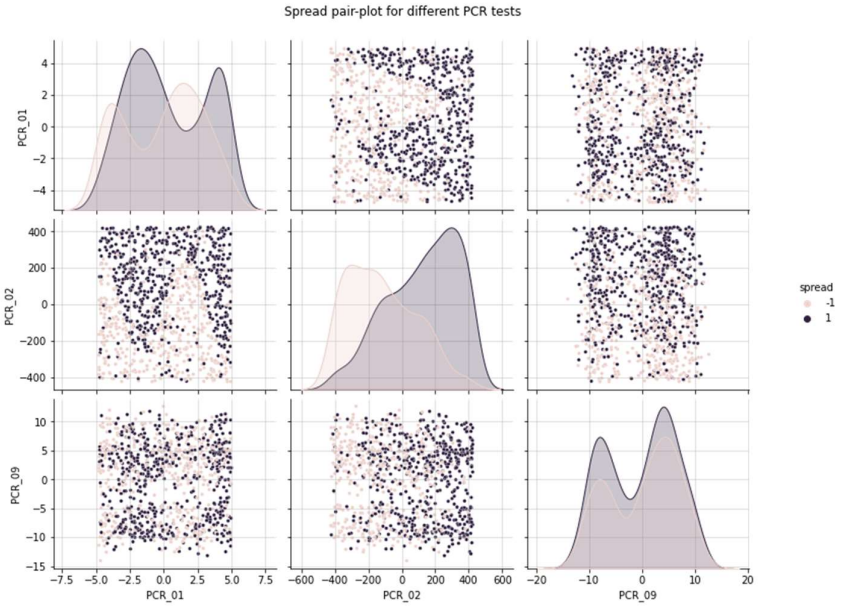
Basic data exploration

(Q5) Compute the correlation between `spread` and each of 3 aforementioned PCR features (see Tutorial 01). Attach the correlations to your report.

```
Correlation between PCR_01 and spread is: 0.081
Correlation between PCR_02 and spread is: 0.504
Correlation between PCR_09 and spread is: -0.039
```

(Q6) Attach the figure to your report.

According to your figure, which 2 of the 3 features are most useful to predict the `spread`?



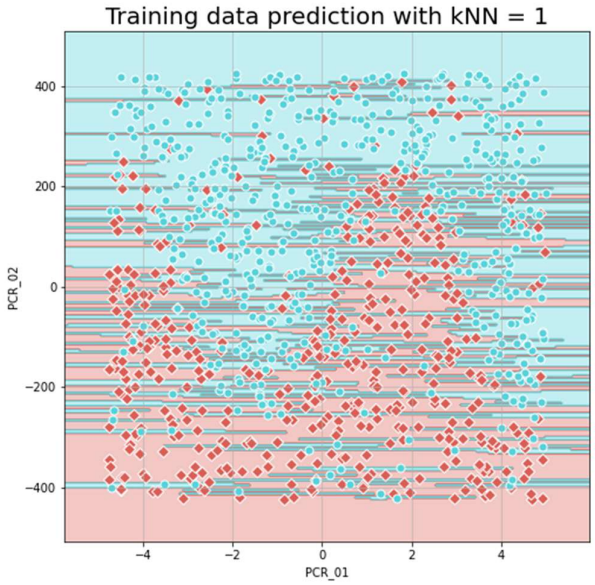
According to our figure, the most useful features to predict spread are PCR_01 and PCR_02. We can tell that by the clear separation of the points that belong to the two different types of spread, which form 2 different clusters.

(Q7) 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.)

Accuracies:

```
Model's training accuracy:  1.0
Model's test accuracy:  0.752
```

Figure:

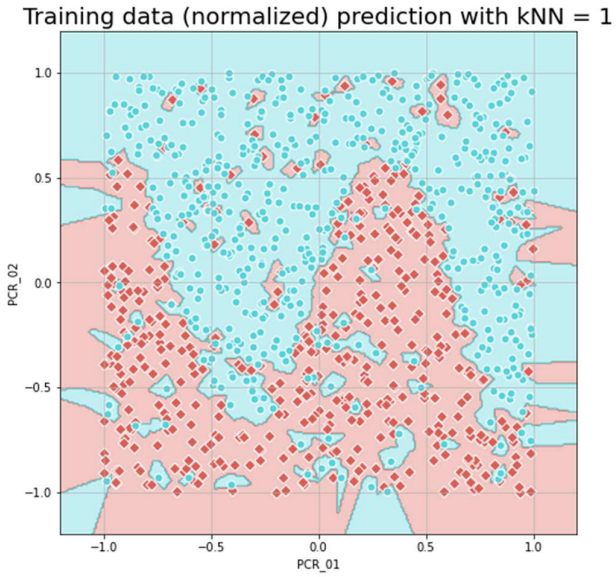


Data normalization

(Q8) 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 normalized 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 **(Q7)** for the same $k = 1$ model on the raw data. Use these results to explain why normalization is important for nearest neighbor models.

Figure:



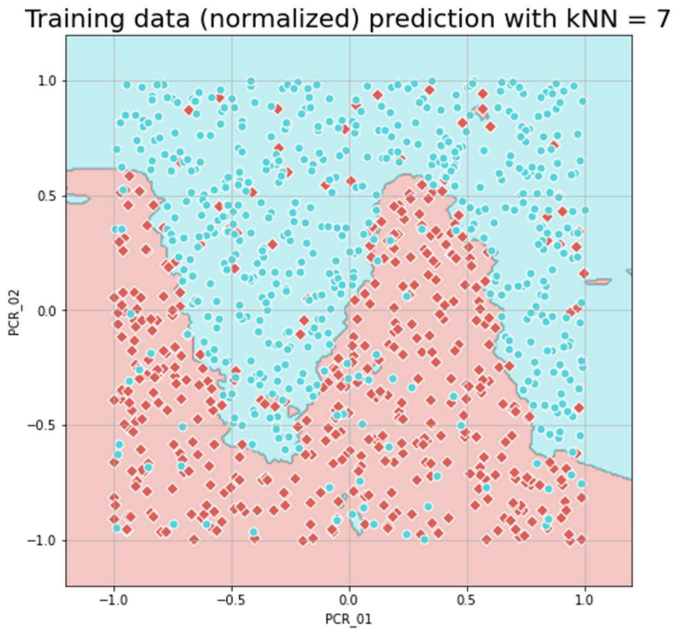
Accuracies:

```
Model's training accuracy:  1.0
Model's test accuracy:  0.824
```

As we can see, the test accuracy of the model is better (0.824 in comparison to 0.752) after normalization and the borders of the blue/red territories in the graph are rounder (rather than sharp edges) and have relatively equal radial spread. It shows us that the normalization is important for nearest-neighbors models, as the normalization balances the significance of the distances between values over the two different axes – In the first graph the values of the "PCR_01" axis have a range of approximately -5 to 5, whereas the values of the "PCR_02" axis have a range of approximately -400 to 400, so the significance of the values of the second axis is larger here. In contrast, the second graph presents an equal range of values for both axes, thus the aforementioned significance is equal ("balanced") in this case.

(Q9) Using the normalized dataset, train another kNN model with $k = 7$. 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 **(Q8)**. Use these results to briefly explain the effect of k on the decision regions.

Figure:



Accuracies:

```
Model's training accuracy:  0.873
Model's test accuracy:  0.888
```

Relative to the Question 8 graph, we can see that the new graph territories are less abrupted. Meaning there're less territories, but the territories are larger.

We know that the larger k value is, the more neighbors we consider for each points prediction. Therefore, in larger k value we'll look at a larger environment for each point. This causes for less abrupted ("smoother") red/blue territories, because singular points in other class territories are influenced by the overall majority, and not by a few singular points.

(Q10) Assume one of the features in a dataset is randomly sampled (i.i.d.) from $\mathcal{N}(0,1)$, and assume the distributions of the other features are unknown.
Why is normalizing that normally-distributed feature using min-max scaling a bad idea? (Assume the number of samples is large and think about kNN's performance.)

In standard normal distribution, the center of the gaussian is at 0, and the majority of the samples (around 35% * 2 = 70%) is in the range (-1, 1) (since: $\mu - \sigma = 0 - 1 = -1$ and $\mu + \sigma = 0 + 1 = 1$). Furthermore, we know that the normal distribution isn't bounded, so the samples that have the max and min values over that distribution, can have values with a very large magnitude – so the factor of normalization could be very large as well, even though the number of occurrences of these extreme samples in very small in comparison to the most samples, which are around (-1, 1). Due to this large factor, the normalization will concentrate most samples in a tight environment of 0, such that the significance of the distance between the samples over this feature will get much smaller in comparison to other features, which will lead to lower accuracy of the kNN results.

In some cases, we could even shift the data from its center, using min-max normalization. If the max is much larger than the min (in absolute value), then the normalization will shift the data towards the max size. The data's midpoint (the normal distribution expected value) will be shifted towards the positive area. Meaning the data will be represented incorrectly.

(Q11) The `blood_type` feature has several categories that are combinations of A, B, AB, O characters and the +,- symbols. These string values are meaningless in terms of separation. A common solution is to use one-hot-encoding (OHE). This is a bitmap indicating the one single category to which the sample belongs. For instance, three categories ("X", "Y", "Z") are encoded by three Booleans into (001,010,100).
How many Boolean features are needed to create such a representation for the `blood_type` feature?

We need 8 Boolean features for the blood_type representation, because we got 8 blood types, and need a different boolean feature for each blood type.

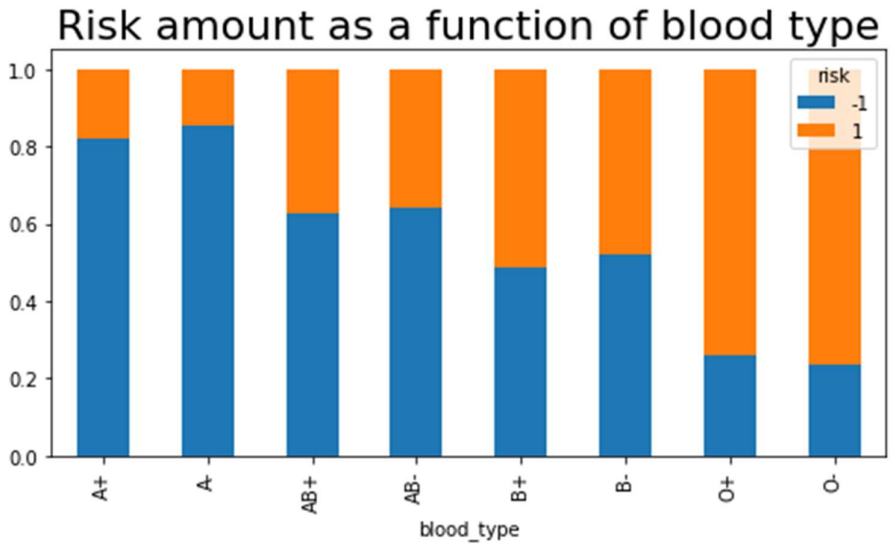
(Q12) Like we did in Tutorial 01 (in code-block [17]), plot a normalized cross tabulation plot (`crosstab`) of the `risk` (not `spread`!) target variable (y-axis) and the different blood types (x-axis).

Don't forget to have suitable titles, axis labels, and grid lines in all your figures!

Attach the plot to your report.

Based only on this plot, how would you partition the blood types into three blood groups? (Specify the groups and briefly explain your logic.)

There might be more than one correct partition.



We would partition the blood types in the following order:

Group 0- O plus and O minus types.

Group 1- B plus and B minus types.

Group 2- All other types.

We divided the groups according to the majority of the risk level in a certain blood type. As we can see in the graph, group 0 has a majority of positive risk, group 1 has approximately even amount of positive and negative risk, and group 2 has a majority of negative risk. By this division, we get a correlation between the blood type group, and the majority of the risk (negative, positive, and even).

(Q13) In **(Q3)** you found features that are neither categorical nor continuous. One of them should have been `symptoms` because it holds string values with multiple categorical values per entry. Can we extract information from this feature that may be useful for our prediction task, i.e., can we craft new features using the `symptoms` feature that are more informative? If so, add these newly crafted features to your `DataFrame` and briefly describe the extraction method you used in your report. If not, explain why that is (2-3 sentences).

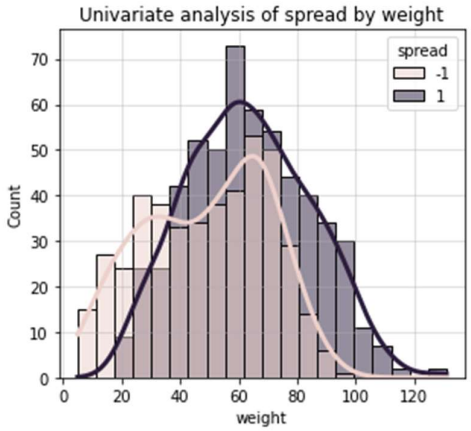
We can craft new features using the "symptoms" feature, that are more informative, by extracting all the reported symptoms from the data strings in this column, and for each symptom we'll create a new Boolean column, where 1 indicates that the patient had that symptom and 0 indicates the opposite. The different symptoms we extracted from the dataset are sore throat, low appetite, shortness of breath, fever, cough. Our method includes scanning of all the symptoms strings (and splitting according the ';' character) of the original dataset and creating unique corresponding columns where we marked 1 for each entry, in which the symptom appeared in the string.

Task:

The other features we transformed and crafted new features for are sex, blood_type, current_location, pcr_date. For sex we transformed "M" values (which represent male) to 1 and "F" values (which represent female) to 0. For blood_type we transformed the column according to the OHE method, using the 3 groups we defined in Q12. For current location we crafted 2 new columns – the first for the x-location and the second for the y-location. For the pcr_date we crafted 3 new columns – the first for the day, the second for the month and the third for the year.

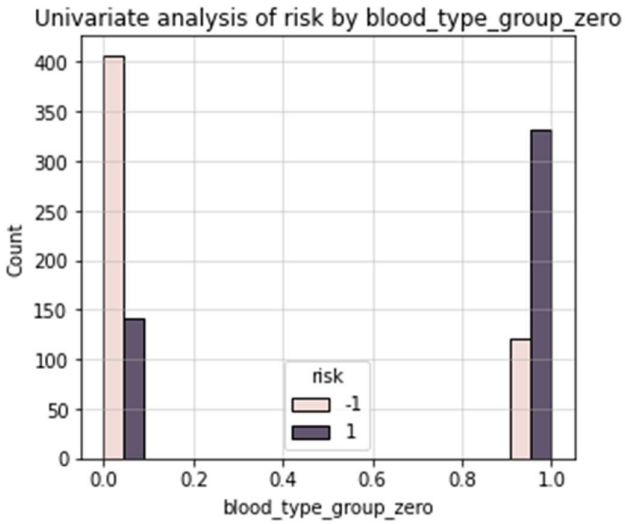
(Q14) According to the univariate analysis, name one feature that seems informative for predicting the `spread` target variable (other than the 2 features you chose in Q6). Attach the appropriate univariate plot and briefly explain (2-3 sentences) why this plot makes you think that feature is informative.

The plot:



The plot makes us think that the weight feature is informative, because we can see that both spread samples classes have distributions which are approximated to normal distribution, with roughly similar mean and variance. The distribution of the high spread sample class (dark color) has significantly more samples with values above 80 than the low spread sample class, and the distribution of the low spread sample class (light color) has significantly more samples with values below 20 than the high spread sample class – which indicates on a difference between the two distributions and implies that the **weight** feature is indeed informative for predicting the spread target variable.

(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 chose the blood_type_group_zero feature, because the percentage of positive risk within the blood_type_zero_group is a clear majority. Also, the percentage of negative risk within the non-blood_type_zero_group is a majority. This is no suprise, as we grouped the different blood types according to the majority of the risk level (in question 12). Therefore, the blood_type_group_zero feature do give information about the risk level.

(Q16) The following code snippet computes the correlation between `risk` label and the rest of the features. Attach to the report the 10 most correlated features to `risk`.

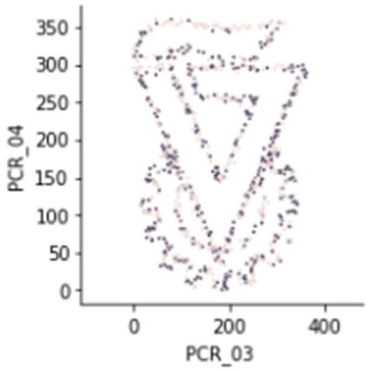
```
risk                    1.000000
blood_type_group_two    0.492880
blood_type_group_zero   0.475704
shortness_of_breath     0.085330
cough                   0.083534
household_income        0.078862
PCR_06                  0.077971
x_location              0.065928
PCR_09                  0.053347
PCR_03                  0.045069
age                     0.044493
```

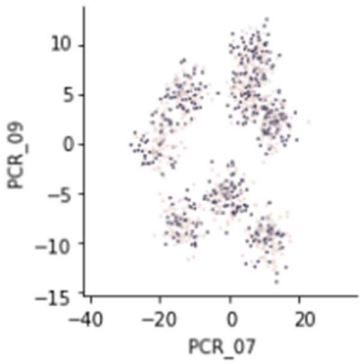
(Q17) Use the above snippet and look at the bivariate analysis for the `PCR` features. Choose two pairs of features that (together) form "interesting" structures (with or without regard to the `risk` variable).

Specify these pairs of features and copy their joint scatter plots to your report. Does any of these pairs (by itself) seem to explain the `risk` variable to a large extent?

We chose the pair PCR_03-PCR_04 (**on the left**) and the pair PCR_07-PCR_09 (**on the right**).

The bivariate analysis for the PCR_03-PCR_04 and for PCR_07-PCR_09 pairs:





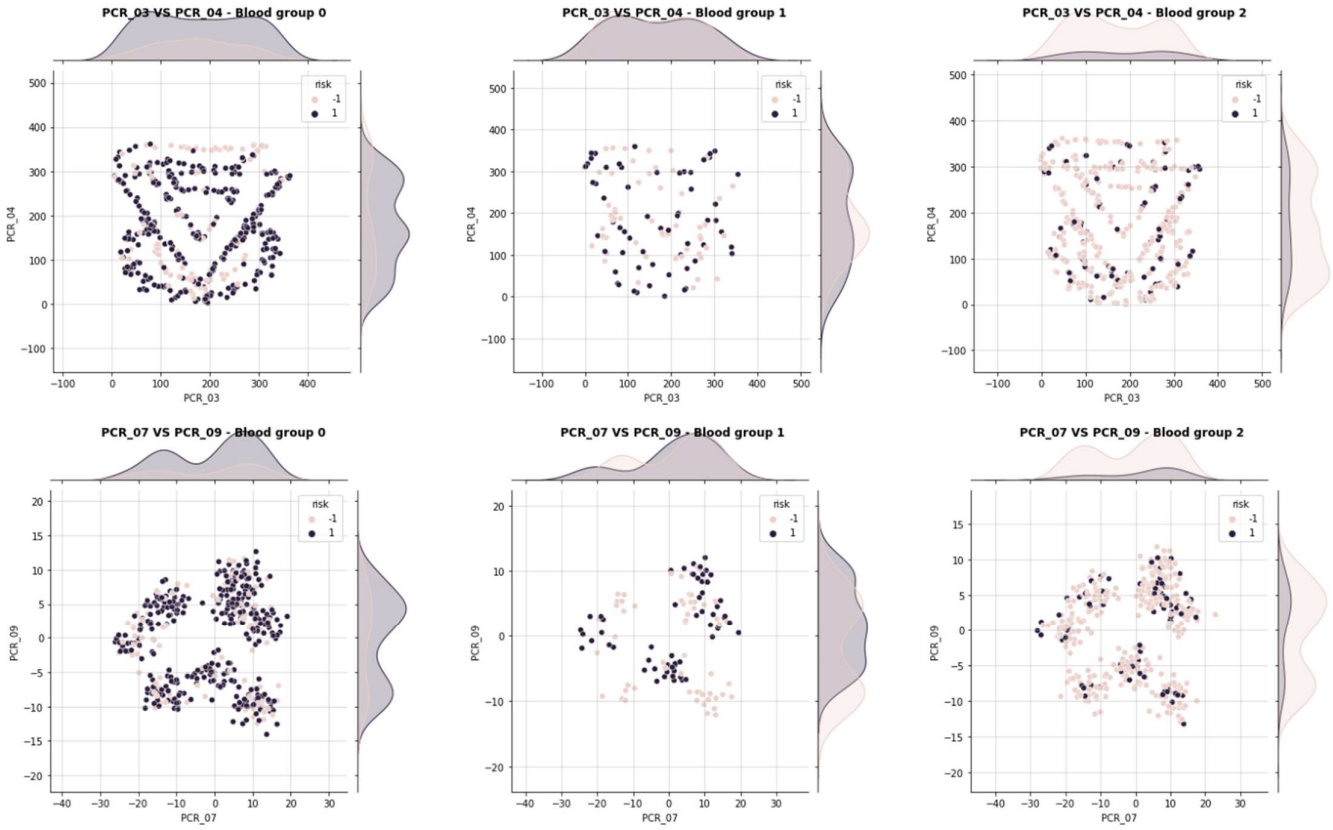
These pairs don't seem to explain the risk variable to a large extent, as the low-risk and high-risk samples (light and dark dots) are mixed and spread along the plots – hence we don't have a clear separation of the different classes using any of the pairs.

(Q18) For each of the two feature-pairs you chose in the previous question, create three `jointplot`s (see Tutorial 01), one for each blood group you created in **(Q12)**, conditioned on the `risk` variable.

For instance, if you chose two pairs (PCR_01, PCR_02) and (PCR_01, PCR_03), and the blood groups are A, B, and C; then one of the jointplots you create should show PCR_01 vs. PCR_02, setting the color (hue) according to the risk variable, and showing only data points from blood group A. And so on.

Attach the 6 resulting plots to your report. Remember to have grids, titles, and axis-labels.

resulting plots:



(Q19) Did the features from the informative pairs of (Q18) exhibit a "high" correlation to the risk in (Q16)? Explain why.

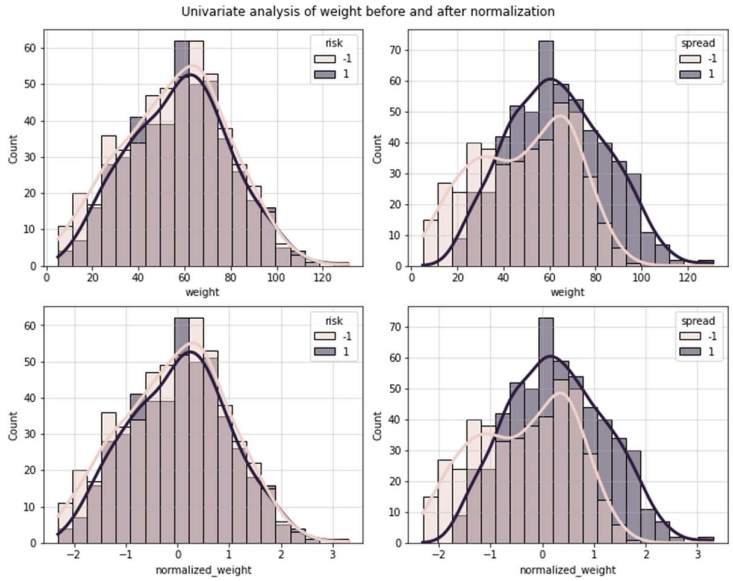
The "PCR_09" and the "PCR_03" features have a higher correlation to the risk according to (Q16), relatively to the other features ("PCR_04" and "PCR_07"), but their correlation values were still very low (0.053 and 0.045 respectively, which are close to 0). Therefore, we deduce that there isn't an obvious correlation between the pairs and risk, at least from the data we have. In the **group 0** and **group 2** joint plots of (Q18), it seems that both pairs we chose do have a "high" correlation with risk, which is reasonable due to the fact we created these groups based on high/low risk amount in the samples.

(Q20) Considering everything that you just saw, which type of model would be most suitable for predicting the risk – kNN, decision trees, or linear model (with no mappings)? Explain your answer in detail (4-6 sentences).

The data points don't seem to differ linearly. Since the linear model separates data linearly, we decided not to use it. For the remaining choice, we prefer kNN, as the data seems to be separated into territories (as seen in the PCR_07 vs PCR_09 blood_group_1 graph). The data does have overlapping areas, but with a good k value (large enough) the overall territory should be recognized. In contrast to the kNN, decision trees are more likely to induce overfitting on the data since we have more parameters to adjust (such as depth and pruning). Furthermore, using continuous features in decision trees might lead to information loss, due to the limitation of using binary trees only. We'll mention that for the number of samples we have – both memory and time complexity of kNN are reasonable. For those reasons, we think kNN is more suitable for predicting the data.

(Q21) In your report, show a univariate analysis visualization before and after normalization for the `weight` feature.

The univariate analysis visualization before (top row) and after (bottom row) normalization:



Part 5: Feature Selection

(Q22) Assume that we have $d_1 \in \mathbb{N}$ features in the dataset, and we wish to find a subset of $1 \leq d_2 < d_1$ features.

How does d_1, d_2 affect the complexity of the number of the models we would have to train when performing (a) forward feature selection, and (b) backward feature selection? Answer for both cases. Ignore other factors (like the time needed to train each model etc.). Explain your answers briefly.

Both in forward and backwards feature selection, we greedily choose features step by step. We'll notate m as the current number of features in the process.

As stated at the sklearn's explanation, we use k cross-validation to do each step. It is also stated that a complexity of $m \cdot k$ model trainings is needed for a single step. k is a constant for the process, therefore, we're going to ignore him for now (as it's affecting all the steps in the same way). Meaning that the complexity of a step is m model trainings.

Therefore, for a forward features selection, we're going to preform:

$\sum_{m=1}^{d_2} m = \frac{(d_2+1)\cdot d_2}{2}$ model trainings, while for the backwards feature selection, we're doing:

$\sum_{m=d_2+1}^{d_1} m = \frac{(d_1+d_2+1)\cdot(d_1-(d_2+1))}{2}$ model trainings. Let's see for which values of d_1, d_2 forward selection is better than backwards selection:

$$(d_1 + d_2 + 1) \cdot \big(d_1 - (d_2 + 1)\big) \geq (d_2 + 1) \cdot d_2$$

$$(d_1)^2 - (d_2 + 1)^2 \geq (d_2)^2 + d_2$$

$$(d1)^2 \geq (d_2 + 1)^2 + d_2 \cdot (d_2 + 1)$$

$$(d1)^2 \geq (d_2 + 1) \cdot (d_2 + 1 + d_2)$$

$$(d1)^2 \geq (d_2 + 1) \cdot (2 \cdot d_2 + 1)$$

$$d_1 \geq \sqrt{(d_2 + 1) \cdot (2 \cdot d_2 + 1)}$$

So, for d_1, d_2 values that maintain the above inequality, we prefer the forwards selection (and the backwards selection otherwise).

(Q23) What are the three features that the selection process found? Do these features include the ones you chose on (Q6)? Do they include the feature you chose manually in (Q14)? If not – are the features found here correlated to the ones from the other questions?

The selection process found the features "weight", "PCR_01" and "PCR_02". In question (14) we chose the weight feature and in question (6) we did chose the "PCR_01" and "PCR_02" features, thus the obtained features indeed include the feature we chose manually. This indicates that the visual data in question (6) was good enough for us to determine which features are better.

(Q24) Generally, why is it important to perform the normalization step before performing sequential feature selection?

Sequential feature selection may get unbalanced results when the data isn't normalized, caused by higher/smaller correlation driven from the value magnitudes of the features (and not from the actual correlation). Making sure the features are normalized, is making sure the data is in the same scale. It's especially important when comparing different features with one another (because we're comparing different scales).

(Q25) Generally, does the choice of a learning algorithm (here we chose 5-NN) matter in a sequential feature selection process? If so, how? If not, why?
Explain your answer in 4-5 sentences.

The choice of a learning algorithm does matter because the sequential feature selection process uses the learning algorithm to predict which features will work best **for the given algorithm**. Different learning algorithms improve on different data types and different data separations. For example, the linear learning algorithm works better on data that is separated linearly. Given a linear learning algorithm, the sequential feature selection process would have chosen features that separate linearly. As we've seen in the previous graphs, the data doesn't really separate linearly, meaning we may have gotten untrusted results and unusable features (for this case at least).

Part 6: Data Preparation Pipeline

(Q26) 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**.
e. **Explanation (reason)**: 1-3 short sentences describing why you made these choices for this feature.

Feature name	Keep	New	Normalization method	Explaination
patient_id	V	X	MinMax	(1)
age	V	X	Scalar	(1)
sex	V	X	None	Old data
weight	V	X	Scalar	(1)
blood_type	X	X	None	Old data
blood_type_group_zero	X	V	None	Already normalized
blood_type_group_one	X	V	None	Already normalized
blood_type_group_two	X	V	None	Already normalized
current_location	X	X	None	Old data
x_location	X	V	Scalar	(1)
y_location	X	V	Scalar	(1)
num_of_siblings	V	X	Scalar	(1)
happiness_score	V	X	Scalar	(1)
household_income	V	X	Scalar	(1)
conversations_per_day	V	X	Scalar	(1)
sugar_levels	V	X	Scalar	(1)
sport_activity	V	X	MinMax	(1)
symptoms	X	X	None	Old data

sore_throat	X	V	None	Already normalized
low_appetite	X	V	None	Already normalized
shortness_of_breath	X	V	None	Already normalized
fever	X	V	None	Already normalized
cough	X	V	None	Already normalized
pcr_date	X	X	None	Old data
pcr_day	X	V	MinMax	(1)
pcr_month	X	V	MinMax	(1)
pcr_year	X	V	MinMax	(1)
PCR_i (i=1,…,5,9)	V	X	MinMax	(1)
PCR_i (i=6,7,8,10)	V	X	Scalar	(1)

(1) MinMax and Scalar normalizations ,were set according to the shape of the data and the expected behavior of the data .Meaning that if the data was centered around some point ,and not too spread ,we tend towards MinMax .Otherwise ,we tend towards Scalar .We also consider the behavior of the data -if we know that the data is bounded, e.g., happines_score which is set by a scale ,we'll use MinMax .If we have unbounded data ,or data that spreads unevenly ,we'll probably use Scalar normalization.