**ML Major 1**

**Submmiters:**

**Husain Azaqy: 213382104**

**Galor Lazar: 325028876**

1. Load the dataset into a Pandas DataFrame.

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

**25 columns**

**1250 rows**

1. 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

I 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.

1. In your report, write a table describing each feature. The columns must be:
   1. Feature name: the name of the feature as it is written in the dataset.
   2. Description: a short sentence with your understanding of the feature’s meaning in the real world.
   3. 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 | Categorical |
| num\_of\_siblings | Number of patient's siblings | Ordinal |
| 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 |

1. [Split](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html) 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]](#footnote-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?

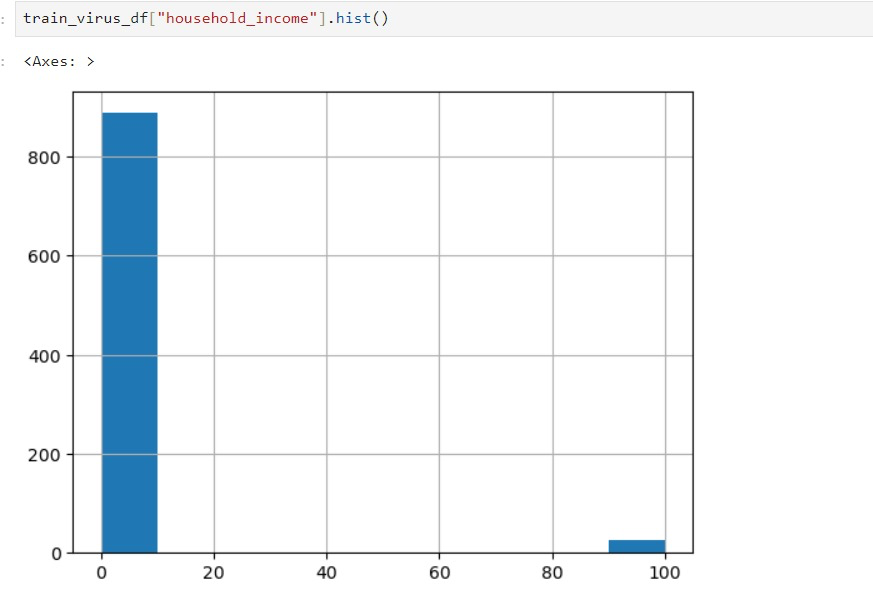
* Since the default split is random, we want the same random selection between multiple analyses to be able to compare them reliably and easily.

1. 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()](https://pandas.pydata.org/docs/reference/api/pandas.isnull.html).

* 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.

1. 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.



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

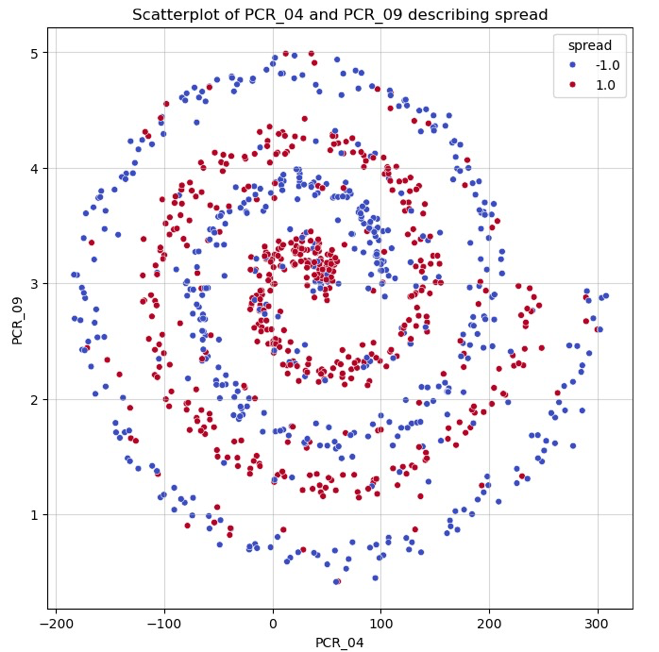
1. For each field where you found missing values, calculate the median and the mean in the training set, **and report it**.

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.

1. 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](https://seaborn.pydata.org/generated/seaborn.pairplot.html) 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).

1. What is the time complexity of the prediction function you wrote, applied on a single test datapoint, in terms of the number of neighbors , the number of training   
   datapoints and the data dimension ? Explain. It is okay to “estimate” the complexity of python library functions. For instance, if you use np.argsort on elements, then its complexity should be . 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:

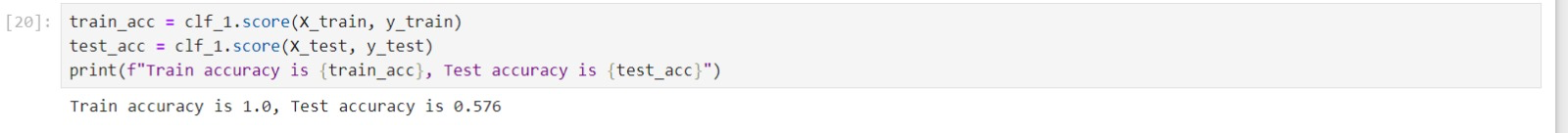
* On average : (quick select) O(m\*d+m+k), in other words: O(m\*d) since k<=m.
* Worst case: (Sorting)) O(mlogm + m\*d +k), since k<=m, we really get : O(mlogm+m\*d)

1. 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.)

A graph showing a spiral of red and blue dots

Description automatically generated



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.

1. Use min-max scaling (between ) to normalize the two features in the temporary DataFrame you created before, and train a new kNN model () 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 **‎(Q10)** for the same model on the raw data. Use these results to explain why normalization is important for nearest neighbor models.

A graph showing a spiral of red and blue dots

Description automatically generated



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,

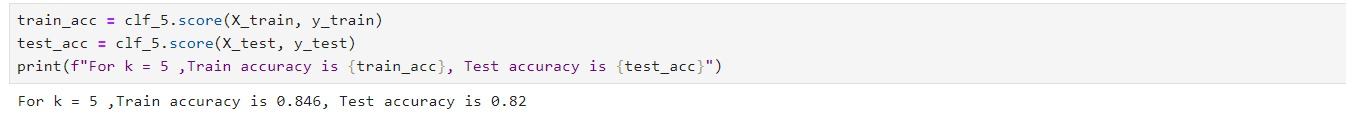
meaning it ensures that each feature contributes equally to the distance computation of the model. When features are on different scales (Like PCR\_04, PCR\_09, the mean of PCR\_04 is 14.2 times larger than the mean of PCR\_09 ), those with larger ranges skew the distance calculations, overpowering the 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.

1. Using the normalized dataset, train another kNN model with . 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 on the decision regions.

A diagram of a spiraling chart

Description automatically generated with medium confidence

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.

1. 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 and the other randomly sampled (i.i.d.)   
   from a chi-squared distribution (see in [Wikipedia](https://en.wikipedia.org/wiki/Chi-squared_distribution)).

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

Why is normalizing both features using min-max scaling to 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 consistently, but it will distort the chi-squared feature's distribution, by taking most values into a small range and overemphasizing outliers. So the normalized won’t be on a comparable scale, which can confuse the model and hurt its performance.

1. [↑](#footnote-ref-1)