HW1 – Data Exploration and Preparation

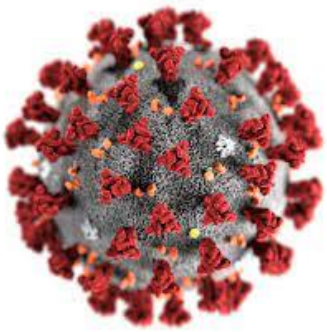
Goal

This exercise is the first of three mini-projects that will guide you in your task of stopping the spread of disease around the globe.

Here we present the **Virus Test Challenge Dataset (VTC)** containing labeled information from patients suffering from all sorts of diseases. Your goal is to understand this dataset and prepare it for prediction.

While we will soon learn several methods for training prediction models, none of them will work without us first observing, understanding, and cleaning our data. So, in this exercise you are asked to perform standard data preparation practices, which will push you towards understanding regularities, and especially irregularities, in your data.

Good Luck!



Instructions

- **Submission**
  - **Submit by:** Sunday, 05.12.2021, 23:59
    - Late submissions will be penalized 5 points per day, up to 3 days.
  - Submissions in pairs only.
  - Submitted on the webcourse.

- **Python environments and more**
  - We recommend using jupyter notebooks. Google colab can be very convenient since it does not require installing anything on your local computer. It will also help you to collaborate with your partner online.
  - Initial notebook here.
    - Demonstrates how to upload a dataset to Google colab and how to download files from Google colab.
    - You can save a copy of this notebook to your Google drive.
  - However, you are allowed to use any Python IDE you choose. For working locally with an IDE, we recommend first installing conda for package management (with Python 3.6 or 3.8), and then installing an IDE like PyCharm or Spyder.

- **Your code**
  - Should be clearly and briefly documented.
  - Variables/classes/functions should have meaningful names.

- **Final report**
  - Should be written in a word processor (Office Word, Google docs, etc.).
    - Should not contain the code itself.
    - Do not submit jupyter notebooks as PDFs.
  - Can be in Hebrew, English, or both.
  - **You are primarily assessed based on your written report.**
  - Answer the questions in this instruction file according to their numbering.
  - Add concise explanations, figures (outputs of your code), tables, etc.
  - You are evaluated for your answers but also for clarity and aesthetics.
  - Tables should include feature names and suitable titles.
  - Plots:
    - Should have suitable titles, axis labels, and legends (if needed).
    - Should have grid lines (except maybe heatmaps).

- **Submit a zip file containing** (please use hyphens, not underscores):
  - The report PDF file with all your answers, named *`id1-id2.pdf`*.
    - Do not include your code in your report.
  - Your code (choose the relevant options for you):
    - Working with jupyter: a notebook with your code, *`id1-id2.ipynb`*.
    - Working with a "traditional" IDE: one clear main script, *`id1-id2.py`*, and any additional files required for running the main script.
    - Data preparation function *`prepare.py`* (from Part 4).
  - Do not submit csv files.

Part 1: Data Loading and First Look

The VTC dataset is available on the course website under the name `virus_data.csv`. You should load the CSV file and explore it using the pandas library. Here you will find many features that are both interesting and relevant for our prediction tasks, along with their ground-truth labels for our target variables: **covid**, **spread,** and **risk**. The binary "`covid`" label determines whether a patient afflicted with COVID-19, the "`spread`" label is a binary score of the patient's potential to spread COVID-19, and the "`risk`" label is a binary score of the patient's risk of being infected by COVID-19. All your decisions in the data preparation process should be made with these targets in mind.

Unfortunately, as with any real-world dataset, VTC also includes many redundancies, missing data, and plain garbage. These irregularities will surely harm your prediction models' performance in future assignments. Throughout this exercise, we will try to minimize these unwanted properties in our features. To do so, we should understand what features are available to us.

(Q1) Load the dataset into a Pandas `DataFrame`.
Answer (in your report): how many rows and columns are in the dataset?

(Q2) Print the `value_counts` of the `num_of_siblings` feature (see Tutorial 01). Add the obtained output to your report. Moreover, describe in one short sentence what you think this feature refers to in the real world. State what type (continuous, categorical, ordinal, or neither) you believe this feature to be. Explain.
Note: Ordinal variables are categorical with a natural order (e.g., year of birth).
Remember to clearly write the number of the question next to your answer.

(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 description (1 short sentence) of the feature's meaning in the real world.
c. Type: Continuous, Categorical, Ordinal, or Other.

NOTE: do not to include the target columns ("`covid`", "`spread`" and "`risk`").

(Q4) For each feature that you marked as type “Ordinal” or “Other” in **(Q2)** or **(Q3)**, explain why you think that is the correct typing for them (no more than 2 sentences per feature).

Part 2: Data Imputation and Cleaning

Partitioning the data

During the learning process, we measure our models’ performance on two disjoint sets: **training** and **test**. A training set is a subset of the dataset from which the machine learning algorithm learns relationships between features and target variables. The test set provides a final estimate of the machine learning model’s performance after it has been trained. Test sets should never be used to make decisions about which algorithms to use or for improving or tuning algorithms.
Note: later in the course, we will use another data subset, called the validation set.

Here we will split our full dataset into a training set containing a random sample of 80% of the data, and a test set containing the remaining 20%.

We will explore why this data partitioning is important later in the course (when we have models to evaluate), but for now the most important thing to remember is that **you may only use the training set when making decisions about the data**.

(Q5) Split the data into training and test sets. Take 80% of the data as the training set and the remaining 20% as the test set. As the `random_state`, use the sum of the last digit 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?

Note: it could be easier for you to answer this question after you completed the rest of the assignment.

[1] i.e. if my i.d is 200033335 and my partner’s is 300011116, then the random state should be 5+6=11

In the following, perform the data preparation actions ONLY on the training set and leave the test set untouched (for now).

Univariate feature exploration

We've already taken a first look at our features and understood their purpose and typing. We now dive deeper into our features. Our first step will be to extract as many features as we can from the data, and then explore every one of them individually.
The learning algorithms we learn in this course often only accept numbers as inputs. Thus, we must transform non-numeric features using meaningful numeric representations.

(Q6) 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 Boolean into (001,010,100). Pandas provides us with this functionality. Use it to generate an OHE to replace the `blood_type` feature. What is the length of this vector?

During this exercise, you will extract new information out of existing features. If that data is categorical (with a reasonable number of categories), consider transforming it to an OHE vector.

(Q7) 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).

(Q8) For each feature that you classified as "other" in **(Q3)**, determine whether useful information can be extracted from this feature. If so, craft new features and add them to your `DataFrame` as you did in **(Q7)**. For every new feature added to the dataset, explain (in 1-2 sentence) why you think this feature is important. Furthermore, transform any other categorical data into OHE vectors.

You will now carry out most of the univariate analysis in your notebook (or IDE). You should not add all the plots to the report, only the ones we specifically request.

For each feature (including extracted features), plot three histograms, one for each target variable. In each histogram use the `hue` keyword to "split" the histogram by the target variable's value (e.g., high/low risk). For continuous/ordinal features you should also use the `kde` keyword to draw the estimated distribution curve (see Tutorial 01).

The following code example generates a 3-column figure of histograms of the features in the `COL_NAME` list. You may use this as a template to generate meaningful plots. Refer to the seaborn documentation to understand more on `histplot`'s keyword arguments. We encourage you to explore these options deeply as they will help you generate informative graphs.

```
COL_NAME = ['blood_type', 'num_of_siblings']

COLS = 3
ROWS = int(np.ceil(len(CAT) / COLS))

plt.figure(figsize=(15, 5 * ROWS))
for i, column in enumerate(COL_NAME, 1):
    plt.subplot(ROWS,COLS,i)
    sns.histplot(data=df, x=column)
    plt.grid(alpha=0.5)

plt.tight_layout()
```

To clarify: in your notebook you should generate 3 histograms for every feature. Each histogram corresponds to one target feature (`covid`, `risk`, `spread`), where the different labels are counted separately and colored differently. Continuous variable histograms should have estimated distribution plots (using the `kde` argument).

(Q9) In your report, add the three histograms you got for `sugar_levels` (only). Describe your findings in detail (5-6 sentences).

Outlier Detection

The VTC dataset might include values that were mistyped (i.e., with a wrong unit of measurement). It might also include extreme cases which might bias our understanding of the phenomena. A common practice in building machine learning pipelines is detecting and removing outliers, sometimes called anomalies, so that they do not skew our learning algorithms.

Please read up on outlier detection here (some implementations here).

(Q10) A *global outlier* is an outlier within its own feature space (e.g., height of 180cm). This stands in contrast to a *contextual outlier*, which seems fine within its own feature space, but is irregular within the context of other features (e.g., a 4-year-old whose height is 180cm).

How can we use our histograms from the univariate exploration stage to find features containing global outlier values? Add one histogram plot to your report that may indicate the existence of such outliers.

(Q11) Render a box plot of `household_income` by the `risk` label and add it to your report. Are the outlier values informative, i.e., do outliers tend to one specific `risk` level over another? If so, how might we use this information to our advantage during prediction? If not, how would we clean the outliers so that they do not skew our models?

Clean all outliers in the dataset. Do this by rendering a box plot for every pair of features that you find relevant for such an analysis (in your notebook).

(Q12) Choose two features in which you found outlier values (you may choose one of your custom features). How did you decide to treat outliers, i.e., what did you do with outlier values, and why? Answer concisely in 2-3 paragraphs (at most 6 sentences per paragraph). Add relevant box plots (or any other informative plot) to your report that support your claims.

(Q13) Extra (not graded but can really improve your performance in future assignments): we suspect the occurrence of outliers is correlated/dependent among PCR features, and with another feature in the dataset. What is the dependence?

You can use your insights in your cleaning process.

Missing data

As you can see, there are many missing values for multiple features. Missing data is a common phenomenon in machine learning that can cause all kinds of problems. There are several strategies for dealing with missing data, including:

- Removing data points (examples) with missing features
  - Very extreme and **should not be done in this exercise**.
- Adding features that indicate '*data is missing here*'
  - Could be useful when the reason the data is missing is informative (such as missing grades for students that do not submit homework).
  - Letting the ML model use it if it wants to, ignore it otherwise
- **Imputation** (the assignment of a value to an attribute by inference)
  - Making assumptions that allow educated estimation of missing values from data.
  - Such assumptions are usually statistical in nature, such as assuming data come from a specific distribution.

In this section, you will substitute missing values with imputed data.

This blog post contains descriptions and examples of basic imputation techniques for both continuous and categorical features.

(Q14) Give one advantage and one disadvantage in choosing median imputation for continuous variables (1 short sentence each).

(Q15) What do we gain when we impute missing categorical data using the "missing category" technique?

(Q16) Analyze `num_of_siblings` and perform imputation on this feature (do not use "missing category"). What imputation technique did you choose and why do you think it is suitable in this case?

(Q17) Impute all missing values in the dataset (do not use "missing category"). In your report, show the univariate analysis visualization as in **(Q9)** before and after imputation for the features `sex` and `weight`.

(Q18) Discuss briefly: what could happen if we impute data before handling outliers?

Part 3: Feature Selection

Inserting non-informative features into our learning algorithms can harm the learning process and introduce unnecessary noise. To prevent this, we commonly look for a subset of features that are most informative for the task we aim to solve. The process of eliminating features is called **feature selection** or **feature pruning**.

Feature selection is a Search / Optimization problem, where the goal is to find an informative feature subset. Such a subset can either be the "optimal" subset, but we often aim for a "good enough" subset. Generally, finding an optimal feature set for an arbitrary target concept is NP-hard. There is a need for a measure for assessing the "goodness" of a feature subset (scoring function) and/or a good heuristic that will (effectively) prune the space of possible feature subsets and will guide the search.

We distinguish between three (very) different techniques to perform feature selection:

- **Filter methods:** Rank feature subsets independently of a classifier using statistical tools we saw in class (uni/bi-variate analysis, correlation, etc.)
- **Wrapper methods:** Use classifiers to assess feature subsets (out of scope).
- **Embedded methods:** Perform variable selection (implicitly) during training (e.g., like in decision trees; also later in the course).

In this assignment, we focus solely on manual filter methods.

(Q19) Add a correlation matrix, like we saw in tutorial 01, to your report and observe the row for `PCR_10`. According to **this row** only, are there any redundant features in our dataset? Explain.

(Q20) Add the histogram plot from the univariate exploration stage comparing `sport_activity` and the target variable "`covid`". According to this plot (only), do you think this feature will help us classify "`covid`"? Explain.

(Q21) Render a jointplot (like we saw in tutorial 01) between `PCR_01` and `PCR_02` where each point is colored according to its "`risk`" label (meaning that 2 points in the plot with different "`risk`" labels should be colored differently). Are both features necessary for classifying "`risk`"? Why or why not.

At this point, you know everything you need to properly clean your data. In the following questions, you will be required to use your newly acquired skills to conduct a meticulous analysis of your data and complete the feature selection and data cleaning processes.

Here are a few tips and guidelines:

- Make sure you handled all outliers and imputed all missing data.
- Do not leave features with values that are not meaningful to the algorithms (e.g., strings).
- Further explore univariate analysis.
  - Play around with seaborn's `histplot` function with different `hue` splits for target and non-target features. Check out the `multiple` keyword (try the "`dodge`" option).
  - Analyze changes in feature distributions before and after normalization.
- Scatter plots in bivariate analysis are often the most revealing
  - Seaborn's `pairplot` is a great starting point for bivariate analysis. This function also has many advanced options that we suggest you explore (especially the `vars`, `hue`, and the `kind` keywords).
  - Play around with seaborn's `jointplot` function with different `hue` splits for target and non-target features and explore other keyword arguments.
    - Use the marginal histograms on the axes to better understand the importance of using both features.

(Q22) Complete the feature selection process using techniques presented in class and in this exercise according to the above guidelines. In your report, describe all interesting findings and your conclusions from those findings. Add plots to support your claims. Your answer should be 1-3 pages long including (reasonably sized) plots.

(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 it is written in the dataset. Names of new features should be meaningful!
b. **Keep**: "V" if the feature is kept, "X" otherwise.
c. **New**: "V" if the feature was handcrafted using other feature(s), "X" otherwise.
d. **Explanation (reason)**: a short sentence describing why you chose to keep/discard this feature.

NOTE: do not to include the target variables "`covid`", "`spread`", and "`risk`".

Part 4: Data Preparation Pipeline

Now that we have finished exploring, cleaning, and pruning our data, it is time to create a data preparation pipeline that will allow us to transform any incoming data point into a "clean" data entry ready for prediction.

Remember that this pipeline **should only reflect the training set**. For example, assuming you imputed some feature with the median value of the training set, then the pipeline should impute missing data from the test set using the same median value, i.e., use the training set median as the "filler value" for all missing data in that feature for all subsets.

(Q24) Write a python module[2] called `prepare.py` containing a single function with the following signature:

```
def preprare_data(data, training_data)
```

The `data` parameter is the dataframe to be cleaned and `training_data` is the training set dataframe used during data exploration. Your function should perform the data cleaning process as summarized in **(Q23)**, including outlier cleaning and data imputation. The output is a copy of `data` (the original parameter should remain unchanged), after it has been cleaned relatively to the provided `training_data`.

If your function uses stochastic steps, make sure to set a seed at its beginning.

You are required to submit `prepare.py`.

Apply the function to both the train and test sets like so:

```
# Clean training set according to itself
train_df_clean = prepare_data(train_df, train_df)

# Clean test set according to the raw training set
test_df_clean = prepare_data(test_df, train_df)
```

Save your two clean dataframes as CSV files and keep them for the next assignment.

Do not submit any CSV files!

[2] If you are using jupyter notebook or Colab and have issues importing external modules, you can simply write the function in your notebook and copy it later to the `prepare.py` file using your preferred text editor. Do not forget to copy the relevant `import` statements that are required for your function to run.