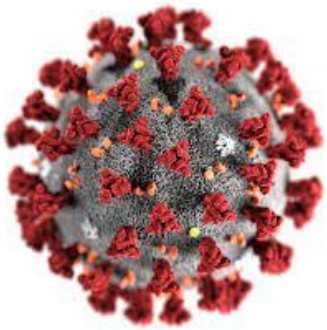
HW1 - Data Exploration and Preparation

Goal

In this exercise, we familiarize ourselves with the first and one of the most important steps in the machine learning pipeline in the real-world. We are talking of course about data preparation and understanding!

While we will soon learn (very) fancy methods for training 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:** 04.05.2021, 23:59
    - Each day of delay in submission costs 5 points.
  - Submissions in pairs only.
  - Submitted on the webcourse.
- **Python environments and more**
  - We advise 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, 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**
  - You are primarily assessed based on a written report.
  - You should answer the questions in this instruction file according to their numbering.
  - Add concise explanations, figures, tables, etc. where needed.
  - Remember 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).

- **Submitted files**

  One zip file containing:

  - The report PDF file containing all your answers.
    - Named *id1_id2.pdf* (with your own ID numbers).
    - Can be in Hebrew, English, or both.
    - <u>Cannot</u> be handwritten.
  - Your code:
    - Working with notebooks: a jupyter notebook containing all your code.
      - Named *id1_id2.ipynb*.
      - If you work on Google colab, then simply use:
        File -> Download -> Download .ipynb.
    - Working with a “traditional” IDE: one clear main python script.
      - Named *id1_id2.py*.
      - Additional files (with classes or functions) which are required for running the main script.

Part 1 - Data Loading

We will be using our very own covid-19 prediction data. This dataset was designed to allow researchers to predict the existence of coronavirus for a given person. The dataset includes many features that are both interesting and relevant for such prediction tasks, but as any real-world dataset it also includes many redundancies, missing data and plain garbage. To be able to successfully predict the outcome variable (covid-19 existence), we first have to understand the types of features that are available to us. Then, before we manipulate the data, we need to split the data such that we can properly evaluate our algorithms.

Concretely, these are steps you are required to do:

1. Load the Virus Test Challenge data from the *virus_hw1.csv* file, which can be found in the course website.
2. Describe what you think each attribute refers to (in a tabular form).
3. Identify and set the correct type of each attribute.
   Note that the three last attributes, i.e. “Virus”, “SpreadLevel”, and “Risk”, will serve as our target labels, which we will predict in the following major assignments.

Part 2 - Data Imputation, Cleaning, Normalization

Partitioning the data

The first step in developing a machine learning model is training and validation. In order to train and validate a model, you must first partition your dataset, which involves choosing what percentage of your data to use for the training, validation, and test sets. Without partitioning our data, our model might learn to overfit on test examples, thus preventing us from properly evaluating its performance.

A training set is the subsection of a dataset from which the machine learning algorithm learns relationships between the features and the target variables. A validation set is another subset of the input data to which we apply the machine learning algorithm to see how accurately it identifies relationships between the known outcomes for the target variables. The test set provides a final estimate of the machine learning model's performance after it has been trained and validated. Test sets should never be used to make decisions about which algorithms to use or for improving or tuning algorithms.

4. Split the data into training, validation and test sets according to patient id.
   A common split for datasets of this type is 60/20/20.
   As the random state, add the last digit of your i.d and your partner's i.d[1].

[1] i.e. if my i.d is 200034545 and my partner's is 300016546, then the random state should be 5+6=11

In the following, perform the data preparation actions only on the training set and apply the resulted transformations to both the test and the validation sets[2]:

Understanding and exploring the data

We start by trying to understand our data. First, please look at the data, identify and set the correct type of each attribute, especially for the ones that were considered as object type, and convert them to the correct type (category, string, float, etc.).
To determine the correct type, recall the first tutorial where we discussed object types and how to analyze them.

5. Identify and set the type of each attribute.
6. For all string typed features (such as "*currentLocation*" and "*Address*"), discuss whether it is useful to modify them and extract useful numeric information from them.
   If you believe this might be useful, generate those features.

[2] label-dependent imputation methods (such as impute by class mean) may not be applied on the test set! Since the test set simulates a case where you don't know the label. For the same reason, statistics (such as mean/median) should not be calculated using data from the test set. Whether to apply such transformation on the validation set is your choice.

Missing data

As you can see in the data, 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 many strategies for dealing with missing data:

- Removing data points with missing features
    - Feels like throwing the baby out with the bathwater
- Adding features that code '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 is from a specific distribution.

To prevent such potential pitfalls, you will need to replace the missing values and substitute them with imputed data. Before doing so, please read up on data imputation (you can do so here or here, but there are many other good alternatives).

7. Decide when to delete features altogether (such as those with too many missing data), which features to impute and why.
8. Fill up the missing values, explain the process you chose to perform and why it was chosen.

Outlier Detection

The Virus Test Challenge data 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, such that they do not skew our learning algorithms. Please read up on outlier detection (for example in here or here. For implementation you can also try this).

9. Generate box plots for features that are suitable for such analysis (in your report choose a few interesting features, do not add dozens of figures).
10. Perform outlier detection using your chosen method for each variable. Explain why you chose each method. Specifically, identify features where there are outliers and present those features using a box plot (hint: one feature for which we know there are outliers is “BMI”). For this feature and for the others, explain how you would automatically identify the outliers.

Data Transformation

The data we often deal with is not organized in an optimal way for learning algorithms. That is, we transform it such that it better captures the information ingrained in it. Most commonly used data transformation methods are:

- Attribute/Feature Construction
    - Create grouping categories – E.g. age groups by year of birth
    - Improve data representation – E.g. log scaling
    - Construct Meta Feature – E.g. contours in an image
    - Code semantic (business, domain) knowledge
- Discretization
    - Binning – Equal-width, Equal-frequency
    - Supervised vs. Unsupervised – Use/Avoid class information
- Categorical to Numeric Conversion
    - Many of the learning algorithms require numeric value representation
- Scaling (standardization, normalization)
    - Control the dynamic range of the feature value
    - Scaled measures ("normalized distance")
    - Avoid numerical instability

While feature construction, discretization and numeric conversion are rather straightforward, data normalization (scaling) requires some statistical understanding. Most common normalization methods are min-max scaling and z-score normalization. Z-score normalization is the process of scaling the data such that it has the properties of a standard normal distribution with mean equal to zero and standard deviation of one. To do that, we simply calculate the Z-score of each observation in the dataset for the feature. In min-max scaling, the data is scaled in such a way that the values range between 0 – 1. In contrast to z-score normalization, the min-max scaling results into smaller standard deviations, which essentially means that we will be suppressing the effects of outliers.

11. Normalize the feature "*steps_per_year*". Decide which normalization method is most suitable and explain why.
12. Please read up on common data transformation methods (for example here or here). Then, for each of the four data transformation methods, identify features that might benefit from such transformation (if there are such features), perform the transformation and explain why it was chosen.

Part 3 - Feature Selection

Inserting non-informative features into our classifier 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. 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's 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.

Like we saw in Tutorial 02, common methods for feature selection are:

- Classification of Feature Methods
    - **Filter methods:** Rank feature subsets independently of a classifier, using correlation between features (and possibly labels).
    - **Wrapper methods:** Use classifiers to assess feature subsets.
    - **Embedded methods:** Perform variable selection (implicitly) during training (out of scope for now).
- Additional considerations
    - **Univariate methods**: consider one variable (feature) at a time
    - **Multivariate methods:** consider subsets of variables (features) together

Recall the tutorial on feature selection methods and especially on sequential methods (for implementation, you can check this out).

13. Generate a correlation table for all suitable features, and analyze correlated features using histograms.
14. Perform the following analysis and explain which criteria informed your decisions. At the very least, aim towards removing one third of the features (a good starting point are features that are highly correlated and might be redundant). This question is a good place to elaborate!
    a. Use at least one wrapper method[3] to remove less informative features.
    b. Perform univariate and bivariate analysis on the selected features to remove at least one feature, and attach the most informative (or cool) scatter plots.
15. Present a table with all features, including an indicator whether they were chosen and a column briefly describing why.
16. Finally, save the selected features in a separate pandas dataframe and split to train, validation and test according to the same split as before. Also save the 3x2 data sets (with all features and selected features) in CSV files.
    No need to submit these csv files, but you should keep them for the next assignments.

[3] Following previous lectures, you should be familiar with kNN, decision trees, and SVM. There are other alternatives, such as linear regression, that you haven't learned yet but are free to use as you see fit.