# Introduction

This rapport summarizes the main points and explains the rational behind my choices in the practical section of the first assignment of the Machine Learning course Fall 2022.

Similar to the practical section, this rapport can be divided into two main sections:

1. Preprocessing
2. Training

Each section is further divided into subsections addressing the corresponding sub-tasks.

# Preprocessing

## Addressing the column 'var7'

The column 'var7' represents date-time data. My work with this column can be summarized as follows:

- fixing the invalid dates: 29-th of february in a non-leap year. My solution is simple: reducing such dates by 24 hours.
- decomposing the date-time data into smaller components: year, month, day, time.
- A straight-forward analysis reveals that:
    1. There is only one year in the entire column: 2019
    2. Only the first 7 months were considered
    3. The distributions of months and days per month are quite similar between the positive and negative classes
- As only the year 2019 is present, this part of the dates is dropped. Additionally the time part was dropped as well for 2 reasons. The dataset is not a time-series dataset. the column 'var3' represent areas meaning the issue is related to large georgraphical areas so it is safe to assume the measurements vary along more signficant time intervals: days, months.
- The datetime data was reduced later to its monthly counterparts as features showed significant variance with respect to (day + month) combination which was represented by the day_of_year temporary feature.
- The month value represents a suitable encoding as the data is not cyclic (only 7 months out of 12) and the number of uniques values is small.

## Encoding

### column 'var6'

For the column 'var6' the Ordinal encoding is a resonable choice as the data is binary: "yes" or "no". As the column's meaning is unknown, no inherited order can be extracted. The machine learning algorithm can either invert or keep the order by assigning negative or positive coefficients.

column 'var3'

I considered a number of encodings for this column:

- One-Hot encoding: a simple and straight-forward encoding. Yet, it introduces 2 main challenges. It signficantly increases the number of features from 7 to more than 200 features. The size increases even more with Polynomial features: it causes memory failures in my local machine with a degree 3 or larger.
- Ordinal Encoding: a simple encoding that does not increase the size of the data. Yet, it also introduces a main challenge: imposing an order on data with no inherited order. Many countries appear more than once assigned with different classes
- Target Encoding: generally defined as replacing the categories with a value computed out of the target variable. I grouped the data by area, and calculated the mean of 'f4' for the corresponding group. Areas that do not appear in the non-NaN data are replaced with the median of 'f4' over all values.

I experiemented with these 3 encodings. Target Encoding resulted in smaller mean squared errors with most of the models. Thus, ***Target Encoding*** was deemed most suitable for numerically representing the 'var3' column. The practical results aligned with the theoretical expectations.

## Imputing 'var4'

The models considered for the sub-regression problem are:

- l2 Regularized Linear Regression: Ridge
- l1 Regularized Linear Regression: Lasso
- Polynomial regression

The regularization hyperparameter as well as the polynomial degree were tuned using the KFold cross validation technique.

To objectively evaluate each model, the non-NAN data was divided into 2 parts: train and test. Each model was tuned using Kfold cross validation and then tested on the test part. The best results were achieved by Polynomial regression with Target encoding.

The practical results aligned with the theoretical expectations as visualizing the variation of 'var4' with respect to other numerical features demonstrate the non-linear relations that linear regression might not be able to catpure to the same extent.

# Training

## Encoding 'var3'

Before proceeding with the training, I needed to re-encode the column 'var7'. As target encoding in the preorcessing phase led to the best results, I decided to use the same technique. As the target variable changed, the actual encoded values need to change as well. Once again I grouped the data by country, aggregated by count and mean. Seeing the target variable as a random variable 'X', I replace each country by its ***conditional expected value*** calculated as the product of number of occurence and the fraction of positive classes out of these occurrences.

# Models

For classification, I trained:

1. Guassian Naive Bayes
2. Logistic Regression
3. K-Nearest Neighbors Classifier

visualizations demonstrated more meaningfull relations between the features and target variable, which was reflected in relatively high performance by all classifiers. The data was split into train and test datasets. Each model was tuned using Kfold cross validation technique and tested on the test data. KNN was the best classifier as it outperformed the other classfiers through 4 performance metrics:

1. accuracy
2. precision
3. recall
4. f1 score: better precision and recall automatically mean better f1 score

Even though the relation between the target variable and numerical features are linear to a certain extent (which explains the high performance of Logistic regression), the relation with categorical feature is not as linear. KNN can capture this interaction (and any type of interactions) due to its distance based approach.

# features

• What were the most critical features with regards to the classification, and why? • What features might be redundant or are not useful, and why?

Both these questions can be answered simultaneously. We can divide the 7 features into 3 main categories:

1. most revelant, (critical): f1, f5
2. moderately relevant : f2, f3, f4
3. little to no relevance: f6, f7(month)

This is supported by 3 different arguments:

1. scatter plots: for numerical continous features, box plots: for categorical ones
2. correlation matrix: visualized with a heatmap
3. having the same or better performance for all classifiers across all performances metrics after dropping ['f6', 'f7']

# PCA

Principle component analysis led to a slight decrease in performance. This can be explained by the small number of initial features. As PCA focuses on building new features (components) that maximize variance, the interactions between the target variable and low-variance features are generally lost. The low variance elements might not be as relevant. However, removing them might lead to a decrease in performance: mainly when the initial performance is higher than 95 % accuracy-wise.

# Multi-Label Learning

1. Multi-label learning problems are problems where each sample / sample is associated with at least two target variables. In other words, the machine learning algorithm will not predict a single value but multiple ones.
2. One possible way to transform this problem into a multi-label classification problem is by considering 'var6' as a second target variable. Both features are binary and each combination of is present in the dataset with relatively the same proportion, which makes it an attractive and resonable option.
3. The current models are not capable of tackling this problem as they are not designed accordingly. My research concerning this issue revealed the following approachs to overcomes the limitations :
    - use more sophisticated models such as Deep Neural Networks, or models provided by the more advanced libraries such as Scikit-Multi-Learn
    - Assuming a n label classification problem, Use problem transformation techniques:
        1. binary relevance: Consider each of the different labels as a single label classification problem, and then for the final multi-label classification, we run n models(each trained on a the same dataset considering only the i-th class) and combine the results: computationally expensive, the correlation between the different labels can be lost.
        2. Use models' chaining: where the first model is trained on the initial dataset to predict the first label/target, the model's outputs is passed along with the initial dataset to the 2-nd model to predict the second label/target, the same procedure continues until the last target. This method preserves the correlation between targets.
        3. Label Powerset: consider every possible combination of the two(or more) variables and treat each combination as a seperate class in the new problem