TASK II (60 marks)

Brief summary of the task

Datasets with complex characteristics, such as data dependencies, high-dimensionality, class imbalance and missing data emerge in most areas of knowledge and always require the data scientist to carefully consider the learning problem and the modelling questions of interest to determine the most adequate approaches to produce generalizable models.

In this task, we will use a specific dataset as a representative example of a challenging modelling problem. The goal is to demonstrate how to handle a high-dimensional dataset – performing exploratory analyses, using the derived insights to prepare the data, and performing modelling and prediction. In the process, you will showcase some of the skills you acquired throughout the unit –data visualisation and manipulation, supervised learning, and statistical thinking.

The details of the application from where this dataset has emerged are not important for the completion of the task. The data happens to have come from a study on which portions of a protein in a certain parasite give rise to immunological responses[1], but datasets with similar properties could have emerged from, e.g., a financial application with dependent indices, a set of monitoring sensors in an engineering plant, or a study of business risk factors in a given sector or country. As mentioned earlier, understanding of the context from which this data has been generated is not required to complete this task. **Focus instead on the actual task of completing the activities as described in this section)**.

Important: You must use the provided R markdown file **Task02.Rmd** to enter your solutions. This is the only file you will submit for this task. If you submit your solution in any other format it will not be marked.

Important: For this task, you must submit the following files:

- **Task02.Rmd**, containing your solution to this task.
- **Task02.pdf**, the PDF produced by “knitting” the solutions file Task02.Rmd.
- **mypreds.rds,** containing the final predictions generated by your model on the “new” data (see Task02.Rmd for details)

Consistency checking: Since Task02 is a more “open” task, it is impossible for us to provide a checkfile. Make sure to carefully follow the instructions in both the description of the task (in this document) and in the Task02.Rmd template.

[1] Ashford, J Reis-Cunha, J; Lobo, I; Lobo, FP; Campelo, F (2021): *Organism-specific training improves performance of linear B-cell epitope prediction. https://doi.org/10.1093/bioinformatics/btab536*

Task Context

In this task you will explore a dataset related to a complex problem known as *epitope prediction*.[3] You are given a set of data created by parsing and consolidating biological data retrieved from multiple online databases, and already highly preprocessed for this coursework. Your goal is to develop an efficient **predictive pipeline** to address this problem.

The data set that is available for you to build your predictive pipeline is called **df.rds.** It has the following structure:

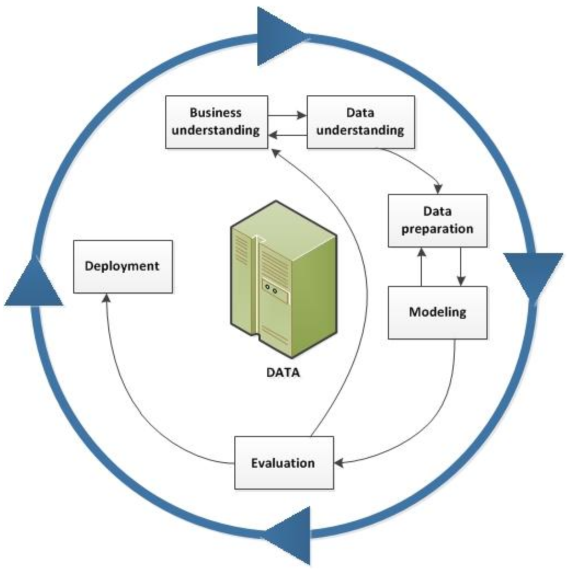
- Two ***ID*** columns named "*Info_PepID*" and "*Info_pos*". Together, these two columns provide a unique reference to each row.
- One ***Group*** column named "*Info_group*"
- Other columns starting with "*Info_*" that are not informative and should be discarded.
- A few hundred ***feature*** columns with names starting with "*feat_*", containing numerical predictive features that were calculated for each observation.
- 1 ***Class*** column, containing the target class.

Important: The data in this task has a characteristic that needs to be considered when modelling: there are dependencies between rows which can break the assumptions of certain pre-processing and modelling approaches. More specifically, observations having the same value of variable "Info_group" are considered to be grouped and, consequently, need to be kept together when data is split (e.g., in cross-validation or train-test splitting) to prevent data leakage.

Besides the data set available for your pipeline development, there is also one ***final validation data set*** (file ***df_holdout.rds***) that is provided. This file has the same structure as the training data, except for the fact that ***the Class attribute is not provided in this set***. It is part of your task to develop a competent data mining pipeline capable of predicting it.

Activities

To adequately build and deploy your predictive pipeline, you will go through several phases of a common data mining framework known as CRISP-DM, which is illustrated below.



(Source: IBM Corporation)

Due to the limitations of this coursework format, you will skip the first one (business/domain understanding), which is usually the most time-consuming. The activities for all other phases are indicated below. For all preprocessing and modelling steps, it is *strongly recommended* that you use the **tidymodels** package.

You will submit a complete summary of your work on the activities above (including both code and discussions) using the submission document **Task02.Rmd**. In that file you can also find additional information about each step.

Overall, it is expected that this task will consist of no more than 1,000 to 1,500 words, or possibly even less (not counting the code blocks). Details and general guidelines of what to write as you document each step are provided in **Task02.Rmd**.

1) "Data understanding": Exploratory Data Analysis (EDA):

In this step you will perform the exploratory data analysis (EDA) of the main dataset. EDA is essential to gain a general understanding of data characteristics such as the size of the data (number of observations and attributes), data types, data quality problems (e.g., missing data, repeated observations or variables), and other specific data characteristics (e.g., presence of outliers, large differences in scale). These insights will later inform different types of *data preprocessing*. EDA and data preprocessing are usually done iteratively, but in this task you'll document them separately. There are several R packages that are useful for EDA.[2] However, it is important to be aware that most EDA guides found online (as well as most functions in the packages indicated in the reference given in the footnote) provide functions that only make sense for *low-dimensional data*. For datasets with large numbers of features (such as the one in our coursework) most graphic profiling and tabulating functions will either freeze your machine (when it runs out of memory) or, in the best case, result in hundreds or thousands of tables and plots that do not provide any insights into the data, from the sheer volume of information displayed. To perform EDA in high-dimensional data, you need to approach this step with a clear idea of which data questions your exploration is trying to answer.

Here is a list of activities that you may want to perform for your EDA. **Don't limit yourself to these**: explore other aspects of the data and try to gain a general understanding of what your dataset "looks like", so that you can make informed preprocessing and modelling decisions later.

- Check number of observations, number of variables, variable types and the balance of your Class attribute.
- Check your data for missing values. Besides detecting which observations / variables contain missing values, it is also useful to check if there are patterns on the missing data (rows or columns with concentrated missing values, for instance).
- Check your numerical features for *outliers*. Outliers are values that differ considerably from the bulk of the data and can bias data transformations and models if not treated appropriately. In low dimensional data, outlier detection can often be performed by simple graphical inspection of univariate boxplots, but this does not scale up well with data dimension. Think (and explore the online literature) before choosing a suitable approach.

[2] See, e.g., Figure 3 of Putatunda et al. (2019): *SmartEDA: An R Package for Automated Exploratory Data Analysis*. https://joss.theoj.org/papers/10.21105/joss.01509

- Check the scale of your numerical features. This is something that is quite easily done for low-dimensional data but needs some thinking for high-dimensional cases. Plotting a table of all maximum and minimum values is impractical - a table with hundreds of rows is unlikely to be very useful. Instead, find ways of checking variable ranges that scale well with the number of attributes. (e.g., plot the maxima and minima of all variables and check visually for large differences of scale). Variable scales will be treated later, as part of data pre-processing.

2) "Data preparation": Data preprocessing:

The insights gained during EDA will inform different *data preprocessing* activities. One thing to keep in mind is that most data preprocessing steps represent *learned transformations* of your data –as such, they need to respect the isolation between training and test/validation sets to prevent data leakage. This step is therefore a good place to isolate a fraction of your available data for your final performance assessment (and keep it isolated until the end of your model development).

Here is a list of activities that you may want to perform for your data preprocessing. **Don't limit yourself to these**: if your EDA uncovered other data issues that need to be addressed, you should follow the insights you got from your data.

- Split a portion of your data (anything between 10 and 25% of your available data should be sufficient) and keep this *test set* isolated until the final assessment step. This will allow you to better estimate your expected generalisation performance. Remember that your data is grouped by *Info_group* (see *Task Context*) – take this into account when doing any data splits. In the code block where you split your data, make sure to explicitly set the seed of your random number generator (using `set.seed(MY_STUDENT_ID)`) so that your splitting is reproducible.

- On the remaining data:

  - If your data has missing values, deal with those using any appropriate approach.

  - If your data has outliers, deal with those using any appropriate methodology (Winsorisation is suggested, but not strictly mandatory).

  - After you deal with any eventual missing values and outliers, scale your features (if needed) so that they are all defined in some common interval.

Note: If you tried several different pre-processing ideas, make sure to state clearly which ones will be ultimately used and which ideas will be abandoned. More instructions are provided in **Task02.Rmd**.

3) Modelling:

In the modelling step you'll deal with tasks that include not only the model fitting itself, but also dimensionality reduction, hyperparameter tuning, and wrapping all your data steps into a well-structured pipeline.

- Based on your pre-processed data so far, fit a preliminary predictive model. Make it simple – a basic decision tree or logistic regression is sufficient. This step is mostly for you to make sure that your data is ready for modelling.

- Use a *pipeline*-type structure to encapsulate your pre-processing transformations and your basic model. If you are using tidymodels (recommended), this can be done using **workflows** (see the tidymodels documentation for details).
- Add a dimensionality reduction step to your pipeline (see, e.g., Tidy Modeling with R).
- If needed, add a strategy to deal with class imbalance (either as a preprocessing step or in the form of specific modelling parameters).
- Replace your basic model in the predictive pipeline by a more sophisticated one (test a few different options and see which one seems better).
- Fit your pipeline and estimate the model performance using an appropriate cross-validation methodology. Use the F1 score as your primary performance metric.
- Use an appropriate hyperparameter tuning strategy to optimise *at least* two hyperparameters of your pipeline.

4) Assessment:

In this step, you will estimate the generalisation performance of your fine-tuned pipeline.

- Apply your fine-tuned pipeline to the test set you isolated in step 2. Check that there are no errors or warnings when you try to generate predictions for this "previously unseen" data. If there are any errors/warnings, address those before proceeding.
- Use the class labels of the test set observations and the predicted classes to estimate the final F1 score of your predicted pipeline. This is the estimated performance of your proposed predictive pipeline on new, unseen data.

5) Deployment:

Finally, use your pipeline to generate predictions for "new" data:

- Load the data file **df_holdout.rds.**
- Apply your pipeline to generate predictions for all observations in **df_holdout.**
- Save your predictions to a file (specific instructions provided in **Task02.Rmd**)