1. Building the Model

**Instructions**

The following task has you develop a machine learning model to predict whether a Free Plan user would convert to a paid subscriber or not.

Think of the features that could help determine the outcome. Share your results by supporting them with relevant metrics, a confusion matrix, or another industry-standard method for model evaluation.**Note that this classification problem deals with a heavily imbalanced dataset.**

**Optional Instructions**

The following instructions refer to Python’s **[imblearn](https://imbalanced-learn.org/stable/references/index.html" \t "_blank)** and **[sklearn](https://scikit-learn.org/stable/" \t "_blank)** libraries. The former is used for re-sampling imbalanced data points, while the latter is applied when we perform the machine learning part.

A list of useful tools and recommended libraries include:

* Python 3.8.8
* imbalanced-learn 0.9.1
* numpy 1.20.3
* pandas 1.3.4
* scikit-learn 1.1.2

You are encouraged to play around with the example solution described below, fiddle with the parameters, try out new approaches, think of different features that could be extracted from the data, etc. **Use the instructions as inspiration rather than a solution.**

* First, consider the features that could be used as a defining difference between users who are likely to subscribe and those who are not, including (but not limited to):
  + Minutes watched on the platform
  + Engagement with quizzes
  + Engagement with exams
  + Engagement in the Q&A hub

You can evaluate the metrics for a fixed period. Our experience has shown that students generally convert within 1 or 2 weeks after entering the platform. Therefore, it is reasonable to confine the features to within several days after registration—for example, minutes watched *N* days after registration.

* Generate SQL queries that create a table with the following columns as inputs:
  + Minutes watched on the platform (Decimal)
  + Number of days in which a student was engaged with the platform (Integer)
  + Engaged with quizzes (Boolean)
  + Engaged with exams (Boolean)
  + Engaged with the Q&A hub (Boolean)

And the following column as a target:

* + Subscribed (Boolean)

**Note:***You can refer to our ‘SQL’ course, which covers queries in depth.*

* Export your table as a .csv file.
* Switch to Python and read the .csv file using, for example, the **pandas** Apply any preprocessing needed.

**Note:***You can refer to our ‘Data Cleaning and Preprocessing with pandas’ course, which teaches you about reading and manipulating data with pandas.*

* Split your data into training and testing sets.
* Choose an approach that helps us handle the imbalance of the data. Choose an [over-sampling method from the imblearn library](https://imbalanced-learn.org/stable/references/over_sampling.html). Fiddle around with the parameters—most importantly with **sampling\_strategy**. Try to understand how this parameter changes the number of data points from each class.

**Note:***Remember to apply the over-sampling only to the training dataset! The test dataset is reserved for testing the model’s performance on data that the model has never encountered.*

* In general, applying an under-sampling technique right after over-sampling increases the model’s performance. Choose an [under-sampling method from the imblearn library](https://imbalanced-learn.org/stable/references/under_sampling.html) and apply it on you over-sampled data. Fiddle around with the parameters—most importantly **sampling\_strategy**. Try to understand how this parameter changes the number of data points from each class.
* Choose a machine learning model to work with. (You can test the performance of several different models.) Feed the re-sampled data into the algorithm.

**Note:***You can refer to Module 3: Machine and Deep Learning on our platform, where we’ve covered many different machine learning algorithms.*

* Reduce the problem’s dimensionality as much as possible—find any features that might not contribute to the model’s performance and remove them.
* Apply a hyperparameter tuning technique, for example grid search.
* Apply a cross-validation technique to better train your model.
* Use your model to predict the outcome given the features from your test set.
* Construct a confusion matrix that would clearly show how many data points from the test set the model predicted correctly and how many of them were misclassified instead.
* Retrieve and discuss relevant metrics such as accuracy, precision, recall, F1 score, AUC, etc.