**Tabular Playground Series - Aug 2022**

**Introduction:** Logistic Regression is a supervised machine learning algorithm. We will discuss the basics of Logistic Regression and how it works, as well as its applications in real-world scenarios. In the evaluation part recall, precession and f1score. Here, precision, recall, and f1 are the precision, recall, and F1-score of the model, respectively. We use the precision\_recall\_fscore\_support function from scikit-learn to calculate these metrics. The average='binary' parameter specifies that we want to calculate the metrics for binary classification and evaluate the model.

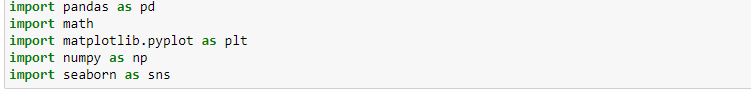
**Data Description:** This data represents the results of a large product testing study. For each product code you are given several product attributes (fixed for the code) as well as several measurement values for each individual product, representing various lab testing methods. Each product is used in a simulated real-world environment experiment and absorbs a certain amount of fluid (loading) to see whether or not it fails.

Your task is to use the data to predict individual product failures of new codes with their individual lab test results.

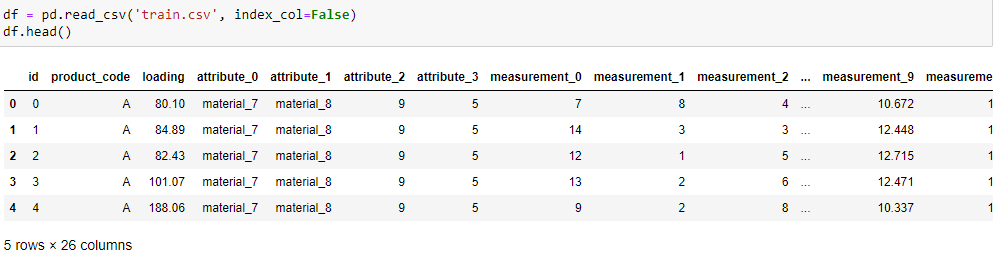
* **train.csv** - the training data, which includes the target failure
* **test.csv** - the test set; your task is to predict the likelihood each id will experience a failure
* **sample\_submission.csv** - a sample submission file in the correct format

# 1. Importing Libraries:

First, importing the important external Python packages using the pip package manager.



1. NumPy is used for mathematical operations like addition, subtraction, multiplication, division, etc. on arrays and matrices.
2. Pandas provides data structures for efficiently storing and manipulating large datasets, and tools for reading and writing data to and from various file formats, including CSV, Excel, and SQL databases.
3. Seaborn is a data visualization library based on Matplotlib which is a plotting library used for creating static, interactive, and animated visualizations in Python.
4. In Python, math is a built-in module that provides access to mathematical functions and constants. The import math statement allows you to use the functions and constants provided by the math module in your code.
5. **Upload the train data and display:**



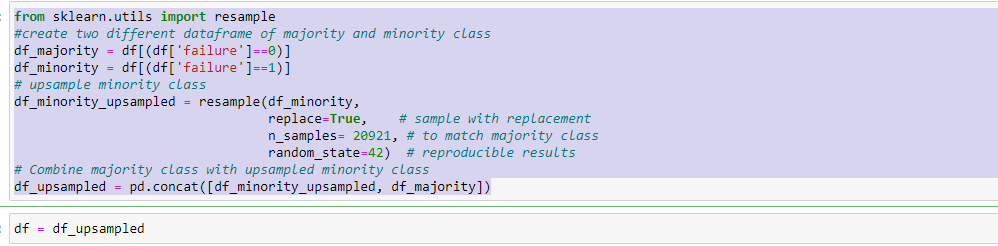
With the help of pandas library, we can read and upload the data in csv form. we can display the first five rows of the data and also by applying **df.shape** we can also find out the number of rows and columns the data has.



1. **Splitting the datasets into two DataFrame:**

This code is performing upsampling of the minority class in an imbalanced dataset using the resampling technique.

The dataset is split into two data frames: df\_majority containing all the majority class samples (where the failure column equals 0), and df\_minority containing all the minority class samples (where the failure column equals 1).



the upsampled minority class data frame is combined with the original majority class data frame using the pd.concat() function, resulting in a new data frame called df\_upsampled that has an equal number of samples for both the majority and minority classes. This new data frame can then be used to train a machine learning model on a more balanced dataset.

1. **Data Preprocessing:**

* At first we need to check whether there is any missing value is present in the data or not. we can code “df.isnull().sum()” to check whether the missing value is present or not.

As in the Notebook we have seen missing value is present we need to fill those missing values and we have filled those missing values with the help of mean values of the data. We have used the code “df1=df.fillna(df.mean())”.

* Performing label encoding is another technique used in data preprocessing. The label encoding process assigns a unique integer value to each category in the dataset.

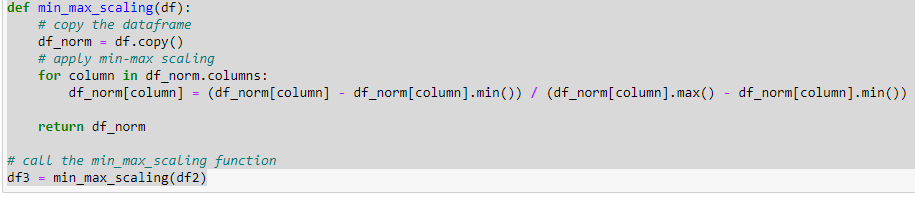


The above code performs label encoding on a categorical variable named 'attribute\_0' in a pandas DataFrame called 'df1’ using the LabelEncoder class from the sklearn.preprocessing module.It can encodes the categorical variables into the numeric ones.

1. **Feature Scaling:**

The significance of feature scaling is that it can improve the performance and accuracy of some machine learning algorithms.

The min\_max\_scaling function takes a pandas DataFrame df as input and applies min-max scaling to each column of the DataFrame. The function creates a copy of the original DataFrame to avoid modifying the original data.



The function applies min-max scaling to each column of the DataFrame by iterating over each column using a for loop, computing the minimum and maximum values for each column using the min() and max() functions, and then applying the scaling formula to each value in the column. Finally, the function returns the scaled DataFrame.

The scaled DataFrame is assigned to the variable df3. Note that the original DataFrame df2 is not modified by this function.

1. **Model Building:**

Logistic Regression is a supervised machine learning algorithm used for binary classification problems, where the goal is to predict a binary output variable.



To import logistic regression in Python, we can use the scikit-learn library, which is a popular machine learning library.

Once we have imported the logistic regression model, we can use it to train a binary classification model on a dataset. Before that, we need to split our data into training and testing sets. By using the code train-test-split.

Now for the validation purposes we need create a variable called pred and then import import precision\_score, recall\_score, f1\_score from sklearn.metrics.

Accuracy: The first line of code computes the accuracy of the model on the testing set. The score method of the trained model takes in the input features (X\_test) and the corresponding true labels (y\_test) and returns the mean accuracy on the test data. In this case, the model has an accuracy of 0.5642, which means it correctly classified 56.42% of the samples in the test set.

Precision: The second line of code computes the precision of the model. Precision is a metric that measures the fraction of correctly predicted positive instances out of all predicted positive instances. The precision\_score function from scikit-learn takes in the true labels (y\_test) and the predicted labels (pred) and returns the precision score. In this case, the precision of the model is 0.5671, which means that when the model predicted a sample to be positive, it was correct 56.71% of the time.

Recall: The third line of code computes the recall of the model. Recall is a metric that measures the fraction of correctly predicted positive instances out of all actual positive instances. The recall\_score function from scikit-learn takes in the true labels (y\_test) and the predicted labels (pred) and returns the recall score. In this case, the recall of the model is 0.5187, which means that the model correctly identified 51.87% of the positive samples in the test set.

F1-score: The fourth line of code computes the F1-score of the model. F1-score is a metric that combines both precision and recall into a single metric. The f1\_score function from scikit-learn takes in the true labels (y\_test) and the predicted labels (pred) and returns the F1-score. In this case, the F1-score of the model is 0.5418, which is the harmonic mean of precision and recall.

For the final prediction part again load the test.csv data and again apply data preprocessing on it and predict the data.