# Real-life Problem #3

## 1. Task

We are given a real life problem, where we have data regarding a credit card customer and 6 actions that they can take, where we will have to predict the probabilities of each action to be taken given a customer. The training data has only one action label for a customer, where other actions are `NaN`. We train using the data where corresponding labels are not `NaN`, and then test using the whole test dataset to get predictions for that action. Doing this for all actions, results in probabilities for all of them. Note that probability in the predictions show the probability of the positive class.

## 2. Implementation

### 2.1. Preprocessing.

We will remove very low correlated features and also convert categorical variables to dichotomous variables.

(1) **Remove the `ID` column**
We do not need this column as it does not contain any information regarding the label. However, we save it during the process to later assign the corresponding predictions to `ID`'s in our output file.

(2) **Fill `NaN` values**
We assign 0 for the `NaN` values. (This does not include `NaN`'s in the labels.

(3) **Remove low correlated features**
We calculate the Pearson Correlation of the features using `corr()` function given in `pandas`. We look at the correlation between features and the target, and remove the features below a given threshold. Note that correlation takes values between 1 and -1, so we only remove those that are close to 0. From a mathematical point of view: Remove feature $f$ if their correlation $c$ holds for $|c| < t$ for some threshold $t$. We have set $t = 0.005$.

(4) **Convert categorical variables to dichotomous variables**
There are 3 features which are categorical. We need to convert them to dichotomous features, and for that we used `get_dummies` by `pandas`.

(5) **Do PCA (Principal Component Analysis)**
To decrease the number of features, we can do PCA. In our example, we will be implementing PCA such that it retains 95% of the variance.

### 2.2. Classification & Posterior Probabilities.

Based on my experiences from previous assignments, I have attempted the best performing methods for the prior tasks in this one too. Excluding SVM, the results were very close, where XGBoost with RF option was the winner with some slightly higher score.

- **Support Vector Machines (SVM)**
  Based on my experience from last assignment where SVM performed well, I again tried to run it. This time, the running time was over 7 minutes so I didn't continue using this method.
- **Gradient Boosting Classifier (GBC)**
  GBC was a promising method for all tasks, and here again it performed quite well.
- **XGBoost (XGB)**
  eXtreme Gradient Boosting is another Gradient Boosting method. We have used this method with and without the `RF` option, where with `RF` option XGBoost uses Random Forest method.
- **Light GBM (LGB)**
  Light GBM is also a Gradient Boosting method, by Microsoft.

We have decided to use XGB with `RF` for our predictions. The cross-validation results for stratified 3 folds are given in table 1.

| Problem | AVG-AUROC | MIN-AUROC | MAX-AUROC | Time (sec) |
|---:|---|---|---|---|
| 1 | 0.69844 | 0.69174 | 0.70700 | 42.01003 |
| 2 | 0.72271 | 0.71267 | 0.73551 | 23.08045 |
| 3 | 0.75821 | 0.75363 | 0.76052 | 39.51755 |
| 4 | 0.80546 | 0.80146 | 0.80975 | 39.19065 |
| 5 | 0.72144 | 0.71717 | 0.72428 | 31.23687 |
| 6 | 0.77677 | 0.76352 | 0.78625 | 23.84741 |

TABLE 1. Evaluation of XGBoost with `RF`