Real-life Problem #2

1. Task

We are given 3 problems, each having a significant number of features and a target label of 1 or 0. The data is about a customer delaying their credit card payment (1 day for problem 1, 31 days for problem 2 and 61 days for problem 3), where the target label is 1 for yes and 0 for no. Unlike earlier assignments, now we have a real-life machine learning problem on our hands. We are given the freedom of trying any method, as long as it works of course. AUROC score (Area under Recieving Operating Characteristic) will be our focus on evaluating our methods. Given the fact that our output variable is numeric, this is a regression problem with more than one feature.

2. Implementation

2.1. **Preprocessing.** We have a lot of features. We will be doing several things to remove some features:

(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.

(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 $t = 0.005$.
As a result, for the problem 1 we remove 10 features, for the problem 2 we remove 36 features and for the last problem we remove 5 features.

(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`. For problem 1 these features are `VAR45`, `VAR47` and `VAR75`, and when they are dichotomous we have 28, 6 and 48 features instead. For problem 2, these features are `VAR32`, `VAR65` and `VAR195`, and when they are dichotomous we have 48, 25 and 6 columns instead. For the last problem, these features are `VAR36` and `VAR153`, where when they are converted to dichotomous variables they become 45 and 3 features instead.

(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. This not only improved the computation time, it also improved the AUROC score. We also **standardize** the data prior to PCA.

2.2. **Classification & Posterior Probabilities.** We have tried a lot of different methods and chose the best performing one overall. We do not predict the class, instead we give the probability of the target belonging to class 1. This also makes sense because in our case class 0 is shown as 0 and class1 is shown as 1, when we think of this in terms of probabilities the same still applies, 0 probability of being 1 means the class is 0 because this is a binary classification problem. Same goes for 1, if class is 1 then probability of being 1 is also 1. In general, the method that performed well on one problem performed welll on other problems, with respect to other methods. We will be be keeping every result for each problem in the tables 1, 2 and 3, which were collected over our evaluations on 3-fold stratified split. Before we start on the methods attempted, here are two honorable mentions:

- **Multinomial Naive Bayes**
  Sounded like a good candidate, especially because as we will later see **Gaussian Naive Bayes** performs relatively well. However, **Multinomial Naive Bayes** does not work with negative values. It is not a good idea to manipulate the data too, so we are not going to use this method.
- **Gaussian Process Classifier**
  Due to its capability of predicting probabilities, I have tried to use this method. However, a single attempt on a fold itself took so much time (over 7 minutes) that I had to interrupt. Given the extent of machine learning today this might seem like a normal time, but other methods dont evevn reach 1 minute in total (over all folds). That is why I have decided not to use this method.

Now, we can begin discussing our methods.

- **Support Vector Machines (SVM)**
  A good candidate for binary classification in high-dimensional spaces is SVM. We are using two options: `gamma='scale'` and `kernel='rbf'`. This seems to be one of the best performing one.
- **Gaussian Naive Bayes (GNB)**
  Given the probabilistic and numeric nature of this problem, we have tried to use GNB. The results were good compared to other methods (excluding SVM). Note that this is extremely fast!
- **Gradient Boosting Classifier (GBC)**
  Also given in the quick and dirty solution by the instructor, we have tried to use this method. This method also yields very good results.
- **Random Forest (RNF)**
  Random Forest is an ensemble method, which is quite famous among

machine learning tasks. Similar to the previous assignment, we have decided to give it a go.

- **Multilayer Perceptrons (MLP)**
  We should never underestimate the power of multilayer perceptrons, so we used it here again.
- **LightGBM (LGB)**
  LightGBM is another gradient booster framwork. It is quite new compared to the rest, and it is famous for being both accurate and fast.
- **XGBoost with RF (XGB)**
  XGBoost is also a boosting framework.

When we compare the results it appears that SVM, GBC and LGB are competing. Below, we are listing the best methods for problems 1, 2 and 3 in order.

- Average AUROC: XGB, XGB, SVM
- Min AUROC: XGB, GBC, SVM
- Max AUROC: XGB, LGB, XGB

Overall it seems gradient boosting classifiers work better, SVM is barely able to win over other methods at the third problem. Also when we look at LGB versus GBC, LGB is just very slightly behind GBC. Based on these results, we have decided to use XGBoost, as it is better, especially on the first problem and majorly on the average AUROC score.

Method	AVG-AUROC	MIN-AUROC	MAX-AUROC	Time (sec)
SVM	0.84783	0.83716	0.85451	36.28986
GNB	0.76791	0.76367	0.77070	**0.05317**
GBC	0.86926	0.85839	0.87726	11.15125
RNF	0.84173	0.82754	0.85052	8.63230
MLP	0.83509	0.82185	0.84234	8.59502
LGB	0.87015	0.85810	0.87962	3.70527
XGB	**0.87143**	**0.85997**	**0.88299**	9.20582

TABLE 1. Method Performances - Problem 1

When I submitted the first draft, I had not used XGBoost wirh RF option in this task, but on my second draft I will use it, and it performs better than other algorithms. There are cases where it falls short by very small score (on magnitudes of 0.01) so I preferred to use XGBoost with RF option.

Method	AVG-AUROC	MIN-AUROC	MAX-AUROC	Time (sec)
SVM	0.76209	0.74233	0.77762	19.37345
GNB	0.69384	0.67763	0.70665	**0.04953**
GBC	0.77780	**0.77487**	0.78132	9.47699
RNF	0.72882	0.72349	0.73654	10.01324
MLP	0.75302	0.73629	0.76176	5.21068
LGB	0.78306	0.77411	**0.80048**	3.90060
XGB	**0.78410**	0.75820	0.79721	8.21246

TABLE 2. Method Performances - Problem 2

Method	AVG-AUROC	MIN-AUROC	MAX-AUROC	Time (sec)
SVM	**0.78689**	**0.78509**	0.79006	7.94476
GNB	0.68891	0.66778	0.70154	**0.02692**
GBC	0.78210	0.77727	0.78991	4.62732
RNF	0.72116	0.70689	0.72902	3.40445
MLP	0.78193	0.77428	0.78618	4.25096
LGB	0.78306	0.77411	0.78510	3.27274
XGB	0.78669	0.77860	**0.79217**	4.66778

TABLE 3. Method Performances - Problem 3