**Project Description**

Congratulations! You’ve completed another training platform course. Now is the perfect time to test your skills and solve a new machine learning problem. For this project, you will be on your own.

When you finish, send your work to the project reviewer. You will receive feedback within 24 hours. After that, you will make any necessary changes to your work and send it for a second review.

Usually, this process has to be repeated several times until you get the green light from the reviewer and all the corrections are approved. That’s all part of the job.

Your project will be considered complete once the project reviewer approves it.

# Project description

Beta Bank customers are leaving: little by little, chipping away every month. The bankers figured out it’s cheaper to save the existing customers rather than to attract new ones.

We need to predict whether a customer will leave the bank soon. You have the data on clients’ past behavior and termination of contracts with the bank.

Build a model with the maximum possible *F1* score. To pass the project, you need an *F1* score of at least 0.59. Check the *F1* for the test set. Additionally, measure the *AUCROC* metric and compare it with the *F1*.

[Data source: https://www.kaggle.com/barelydedicated/bank-customer-churnmodeling{target="blank"}](https://www.kaggle.com/barelydedicated/bank-customer-churn-modeling)

## Project instructions

 Download and prepare the data. Explain the procedure.

 Examine the balance of classes. Train the model without taking into account the imbalance. Briefly describe your findings.

 Improve the quality of the model, taking into account the imbalance of classes. Train different models and find the best one. Briefly describe your findings.

 Perform the final testing.

## Data description

The data can be found in /datasets/Churn.csv file. [Download the dataset.](https://code.s3.yandex.net/datasets/Churn.csv)

**Features**

*RowNumber* — data string index

*CustomerId* — unique customer identifier

*Surname* — surname

*CreditScore* — credit score

*Geography* — country of residence

*Gender* — gender

*Age* — age

*Tenure* — period of maturation for a customer’s fixed deposit (years)

*Balance* — account balance

*NumOfProducts* — number of banking products used by the customer

*HasCrCard* — customer has a credit card *IsActiveMember* — customer’s activeness

*EstimatedSalary* — estimated salary

**Target**

*Exited* — сustomer has left

# Project evaluation

We’ve put together the evaluation criteria for the project. Read this carefully before moving on to the task.

Here’s what the reviewers will look at when reviewing your project:

How did you prepare the data for training? Have you processed all of the feature types?

Have you explained the preprocessing steps well enough?

How did you investigate the balance of classes?

Did you study the model without taking into account the imbalance of classes?

What are your findings about the task research?

Have you correctly split the data into sets?

How have you worked with the imbalance of classes?

Have you performed the model training, validation, and final testing correctly?

How high is your *F1* score?

Did you examine the *AUCROC* values?

Have you kept to the project structure and kept the code neat?

You have your takeaway sheets and chapter summaries, so you are ready to proceed to the project.

Good luck!

To build a model with the maximum possible F1 score, we will need to optimize our model's precision and recall simultaneously. Here is a step-by-step approach to achieve this:

1. Preprocess the data: Clean and preprocess the dataset, handle missing values, encode categorical variables, and scale numerical variables if necessary.
2. Split the dataset: Split the preprocessed dataset into training, validation, and test sets. The training set will be used for model training, the validation set for hyperparameter tuning, and the test set for final evaluation.
3. Choose a classification algorithm: Select an algorithm suitable for the given dataset. Depending on the nature of the problem (binary/multiclass), consider using algorithms such as logistic regression, random forest, gradient boosting, or support vector machines.
4. Define performance metrics: Since the project requires measuring both F1 score and AUC-ROC, make sure to define appropriate metrics for evaluation.
5. Train the model: Fit the chosen algorithm on the training dataset and fine-tune hyperparameters using the validation set. Use techniques like grid search or random search for hyperparameter tuning.
6. Evaluate the model: Once trained, evaluate the model's performance using the test set. Calculate and record the F1 score and AUC-ROC for comparison.
7. Iterate and improve: If the initial performance does not meet the minimum F1 score requirement, iterate on steps 3-6 by trying different algorithms, feature engineering techniques, or hyperparameter tuning strategies until the desired F1 score is achieved.

Comparing F1 score and AUC-ROC: The F1 score and AUC-ROC are both common metrics used to assess the performance of classification models. While the F1 score provides a measure of the model's balance between precision and recall, the AUC-ROC evaluates the model's ability to distinguish between classes by considering the trade-off between true positive rate (sensitivity) and false-positive rate.

In general, a higher F1 score indicates a better balance between precision and recall, while a higher AUC-ROC suggests better class separation. However, the interpretation may vary depending on the problem at hand. For example, if class imbalance is significant, the AUC-ROC might be more informative as it is less affected by class distribution.

Thus, both the F1 score and AUC-ROC should be considered together to have a comprehensive assessment of the model's performance.

**To complete the project, follow these instructions:**

1. Download and prepare the data:
   * Identify the source of the data and retrieve it.
   * Analyze the data structure and format.
   * Preprocess the data as necessary, including handling missing values, outliers, and data transformations.
   * Split the data into training and testing sets.
2. Examine the balance of classes:
   * Determine the distribution of classes in the dataset.
   * Assess whether there is a class imbalance, where one class significantly outnumbers the others.
   * Train a model on the data without considering the class imbalance.
   * Describe your findings in a brief summary, including any issues or limitations observed.
3. Improve the model quality considering class imbalance:
   * Apply techniques to address class imbalance, such as oversampling the minority class or undersampling the majority class.
   * Experiment with different machine learning models, such as decision trees, logistic regression, support vector machines, or neural networks.
   * Train these models on the modified dataset.
   * Evaluate the models using appropriate performance metrics, such as accuracy, precision, recall, or F1-score.
   * Compare the results of different models and identify the best-performing one.
   * Describe your findings in a brief summary, including the impact of addressing class imbalance on the model's performance.
4. Perform the final testing:
   * Using the best-performing model identified in the previous step, apply it to the testing set.
   * Evaluate the model's performance on the testing set to gauge its generalization ability.
   * Calculate relevant performance metrics to provide insights into the model's effectiveness.
   * Summarize the final testing results and draw conclusions about the model's performance.

Throughout the project, make sure to document your process, including any data preprocessing steps, model training configurations, hyperparameter tuning, and performance evaluation. This documentation will help explain your findings and justify the final model's selection and evaluation.