# Progress Report 4

In this report, I'll talk about data collection, preparation for modelling and modelling strategy

## Data Collection

In total, I collected 13 datasets for experimenting class imbalance solutions. In Table 1. You can see the list of datasets.

| Dataset_Name | Row_Count | Minority_Class_Percent | Target_Column |
|---|---|---|---|
| Fraud | 284807 | 0.172 | Class |
| Fraud2 | 250000 | 0.5 | is_fraud |
| Wine | 6497 | 3.76 | target |
| Letter-a | 20000 | 3.94 | letter_a |
| Abalone | 1477 | 4.52 | target |
| Pendigits | 10992 | 9.59 | is_9 |
| Sick_euthyroid | 2194 | 10.05 | target |
| Covertype | 581012 | 14.77 | target |
| Letter-vowel | 20000 | 19.39 | is_vowel |
| Contraceptive | 1743 | 22.6 | target |
| Splice-junction | 3186 | 24.07 | target |
| Adult | 32561 | 24.08 | target |
| Churn | 7043 | 26.53 | target |

Table 1. Datasets

As you can see we have datasets with minority class percent between 0.17% and and 26.53%. Some of the datasets had multiclass target so I redefined targets in those case to get binary classification tasks.

Additional to these datasets, I generated synthetic data (with class distribution of 70%/30%).

```
X, y = make_classification(
    n_samples=200000,
    n_features=10,
    n_informative=7,
    n_classes=2,
    random_state=42,
    weights=[0.7,0.3],
    hypercube=False,
    class_sep=0.01,
    flip_y=0.15
)
```

By using this dataset, we can take samples where class distributions are different and do our experiments on the same data. The purpose of this is to ensure that model performances are only changes by minority class percent, otherwise nature of each dataset can affect model performance unexpectedly. Furthermore, by using class_sep and flip_y parameters, I made classification task harder.

You can refer to "Data_Collection.ipynb" notebook in the codes folder to see all details about data collection.

## Data Preparation

For each of the 13 datasets below preparations have been done:

1. Redefining target if needed (done in Data Collection)
2. Encoding the categorical variables
3. Correcting type inconsistencies
4. Imputing missing values if missing value is obvious, otherwise our model (lgbm) will deal with them

I will not go into details of applied preparations to each dataset. You can refer to "Model_Data_Preparation.ipynb" notebook in the codes folder to see all details about data preparations.

## Modelling and Experiment Strategy

LightGBM (Light Gradient Boosting Machine) is chosen as a baseline model for the experiment. LightGBM is an ideal choice as the baseline model in this research project due to its exceptional performance and efficiency in handling large-scale datasets and imbalanced data. LightGBM is a gradient boosting framework that utilizes a tree-based learning algorithm to construct powerful ensemble models. It offers several advantages that make it well-suited for imbalanced classification tasks.

There are 3 main reasons why I choosed LGBM:

1. Boosting models are generally performing better than other statistical models
2. LightGBM doesn't need a lot of data preparation. We don't need worry about data distribution or missing values etc.
3. LightGBM has built in imbalance technique. By simply changing model parameters we can use it and with other balancing techniques we can assess its performance.

## Balancing techniques

1. **Upsampling**:
    - Upsampling, also known as oversampling, involves increasing the number of instances in the minority class by creating synthetic duplicates or generating new samples using various methods (e.g., bootstrapping). By replicating minority class samples, upsampling aims to balance the class distribution and provide more training data for the model to learn from the positive class, thereby improving its performance on the minority class.
2. **Downsampling**:
    - Downsampling, also known as undersampling, involves reducing the number of instances in the majority class by randomly removing samples. The objective is to create a more balanced dataset by reducing the dominance of the majority class. Downsampling can help prevent the model from being biased towards the majority class and enhance its ability to identify patterns in the minority class.
3. **SMOTE (Synthetic Minority Over-sampling Technique)**:
    - SMOTE is a popular oversampling technique that generates synthetic samples for the minority class by interpolating between existing samples. It selects a minority class instance, identifies its k-nearest neighbors, and creates new instances by combining the features of the selected instance with those of its neighbors. SMOTE helps overcome the problem of overfitting that may occur with simple upsampling and can effectively address the class imbalance issue.
4. **BalancedBaggingClassifier**:
    - The BalancedBaggingClassifier is an ensemble method that combines the concept of bagging with resampling techniques. It creates multiple bootstrap samples from the original dataset and trains a base classifier (e.g., decision trees) on each sample. The difference is that each bootstrap sample is balanced, ensuring that the class distribution is approximately equal.
5. **LightGBM Imbalance:**
    - LightGBM has a built-in feature to address class imbalance directly during the model training process. This built-in balancing is implemented through the is_unbalance parameter in the LightGBM library. When is_unbalance is set to True, LightGBM automatically adjusts the class weights based on the ratio of samples in the different classes. This allows the model to assign higher importance to the minority class during the training phase.

In conclusion for each of the datasets (both collected and synthetic datasets) 6 models will be built:

1. LGBM Baseline
2. LGBM Upsample
3. LGBM Downsample
4. SMOTE LGBM
5. LGBM Balanced Bagging
6. LGBM_Imbalance

In the next report I will talk about how models are built technically, experiment results and analysis of them.