Q1 Readme

1.1 Algorithm Introduction

To address the problem of data imbalance, we can analyze it from both the data and algorithm perspectives.

On the **data** level, several approaches can be considered:

- Over-sampling: This involves generating synthetic samples for the minority class to balance the dataset. One method is SMOTE (Synthetic Minority Over-sampling Technique), which creates synthetic examples by interpolating between neighboring minority class instances.
- Under-sampling: This approach involves reducing the number of instances from the majority class to match the minority class. One method is ENN (Edited Nearest Neighbors), which removes instances from the majority class that are misclassified by their nearest neighbors from the minority class.
- Random sampling: This technique randomly selects a subset of instances from the majority class to balance the dataset. However, this approach may result in information loss if the selected subset is not representative.
- Combined over-sampling and under-sampling: Methods like SMOTEENN combine both over-sampling and under-sampling techniques to address data imbalance. SMOTEENN first applies SMOTE to generate synthetic samples for the minority class and then applies ENN to remove instances from both classes that are misclassified.

On the **algorithm** level, the following strategies can be employed:

- Class weighting adjustment: Many algorithms allow you to assign different weights to different classes based on their imbalance. Increasing the weight of the minority class will make the algorithm pay more attention to it during training.
- Algorithm selection: Some algorithms are naturally more sensitive to imbalanced datasets and can perform better without explicit data preprocessing. Random Forest and Gradient Boosting are often mentioned as algorithms that handle imbalanced data well due to their ensemble nature and built-in mechanisms to handle class imbalance.

1.2 Experiment Steps and Methods

Firstly, merge the datasets within the train file to create a training set, called all_data.
Then use SMOTE, Class weighting adjustment and Random Forest in the Q1 solution.

1.2.1 SMOTE+RandomForest

- Data Preparation:
  - The "all_data" dataset was divided into two parts: "train" and "valid."

- The number and proportion of samples belonging to class 0 and class 1 were calculated within the "train" dataset, revealing a significant imbalance.

```
train Label: 0, Count: 102106
train Label: 1, Count: 5277
train Label: 0, Percentage: 95.09%
train Label: 1, Percentage: 4.91%
```

- Data Balancing:
  - Employ the SMOTE (Synthetic Minority Over-sampling Technique) method to address the class imbalance issue in the "train" dataset.
  - SMOTE generated synthetic samples for the minority class (class 1) to achieve a more balanced distribution of samples between class 0 and class 1.
- Model Training:
  - The RandomForestClassifier model was selected for training

```
RandomForestClassifier
RandomForestClassifier(max_depth=5, min_samples_leaf=5, min_samples_split=5,
                       n_estimators=200, random_state=1)
```

  - The model was trained using the balanced "train" dataset.
- Model Evaluation:
  - Evaluated the trained mode on the "valid" dataset

```
Validation Set Evaluation:
              precision    recall  f1-score   support

           0       0.99      0.60      0.75     25550
           1       0.10      0.92      0.19      1296

    accuracy                           0.62     26846
   macro avg       0.55      0.76      0.47     26846
weighted avg       0.95      0.62      0.72     26846
```

  - The primary focus was to achieve a high recall for class 1.
- Model Testing:
  - The trained model was then used to make predictions on the "test" dataset.
  - Generate a classification_report

```
test Set Evaluation:
              precision    recall  f1-score   support

           0       0.98      0.74      0.85      6280
           1       0.14      0.74      0.23       343

    accuracy                           0.74      6623
   macro avg       0.56      0.74      0.54      6623
weighted avg       0.94      0.74      0.82      6623
```

1.2.2 Class Weighting+RandomForest

- Data Preparation:
  - The number of samples and the proportion of samples belonging to class 0 and class 1 were computed within the "train" dataset. A substantial imbalance was observed between the two classes, but no specific handling or processing was done.

- Model Training:
  - The RandomForestClassifier model was selected for training.
  - During training, the class_weight parameter was set to {0: 0.05, 1: 0.95} to address the class imbalance issue. This assigns a higher weight to class 1, indicating its relative importance in the model training process.

```
clf = RandomForestClassifier(n_estimators=200,class_weight={0: 0.05, 1: 0.95},
                             max_depth=10,
                             min_samples_split=5,
                             min_samples_leaf=5,
                             random_state=1)
```

- Model Testing:
  - The trained model was then used to make predictions on the "test" dataset.
  - Generate a classification_report

	precision	recall	f1-score	support
0	0.97	0.85	0.91	6280
1	0.17	0.56	0.27	343
accuracy			0.84	6623
macro avg	0.57	0.71	0.59	6623
weighted avg	0.93	0.84	0.88	6623

References

https://www.cnblogs.com/massquantity/p/9382710.html