Rock Facies Imbalanced Classification with Over-Sampling Techniques

Ryan A. Mardani and Daniel Trad

ABSTRACT

In classification problems, if the dataset is skewed, most machine learning algorithms produce poor prediction results for minor classes. In this study, we used different over-sampling techniques to balance the dataset that includes well logs and rock facies. XGBoost model is employed for this multi-class classification problem. We found that oversampling can improve prediction results in minor classes. Overall, Synthetic Minority Oversampling Technique (SMOTE) is a better candidate for oversampling, though for some classes Adaptive Synthetic Sampling (ADASYN) could compete with the SMOTE performance.

INTRODUCTION

Sometimes real-world datasets are imbalanced. This means that the frequency of classes in the dataset is not similar, resulting in the majority and minority terms emergence. Major classes are those classes that are dominant in the dataset while minor classes show less occurrence in the whole dataset.

Most machine learning (ML) algorithms will ignore minor classes during the training process which will result in poor performance in minority classes. If these classes have special importance, which in practice they have, like a thin layer of rare elements like ore in the mining industry, we need to approach such skewed data with imbalanced classification considerations. Before diving into the methodology, let’s define the imbalanced ranges. If the minor class has less than 1% of the sample population in the whole dataset, it is called extremely imbalanced (Table 1), for the range of 1% to 20% is called moderately imbalanced, and for 20% to 40% is called mild imbalanced dataset.

Table1: levels of imbalance

Degree of imbalance	Proportion of Minority Class
Mild	20-40% of the data set
Moderate	1-20% of the data set
Extreme	<1% of the data set

The dataset that we used for this research is open data published as FORCE 2020 for lithology prediction from well logs in Norwegian offshore. It contains more than 100 wells with common wireline logs and interpreted lithology as facies classes. Well logs are sampled every 0.1520 meters. Some wells have a noticeable number of missing values while others carry rich data points. Figure1 shows the wells location map as well as available data points in each well. The size of the circle is related to the number of data available for that well. There are more than 20 features (physical measurements) available in the dataset, and the target is lithology classes (multi-class classification problem). Figure

2 shows the histogram of target classes on the left as well as numerical information in the table on the right. While shale is the major class in this dataset, the others can be categorized as minority classes with different levels of severity. Sand, Sand/Shale, Lime, Tuff, and Marl are moderately imbalanced. The rest of them are extremely imbalanced.

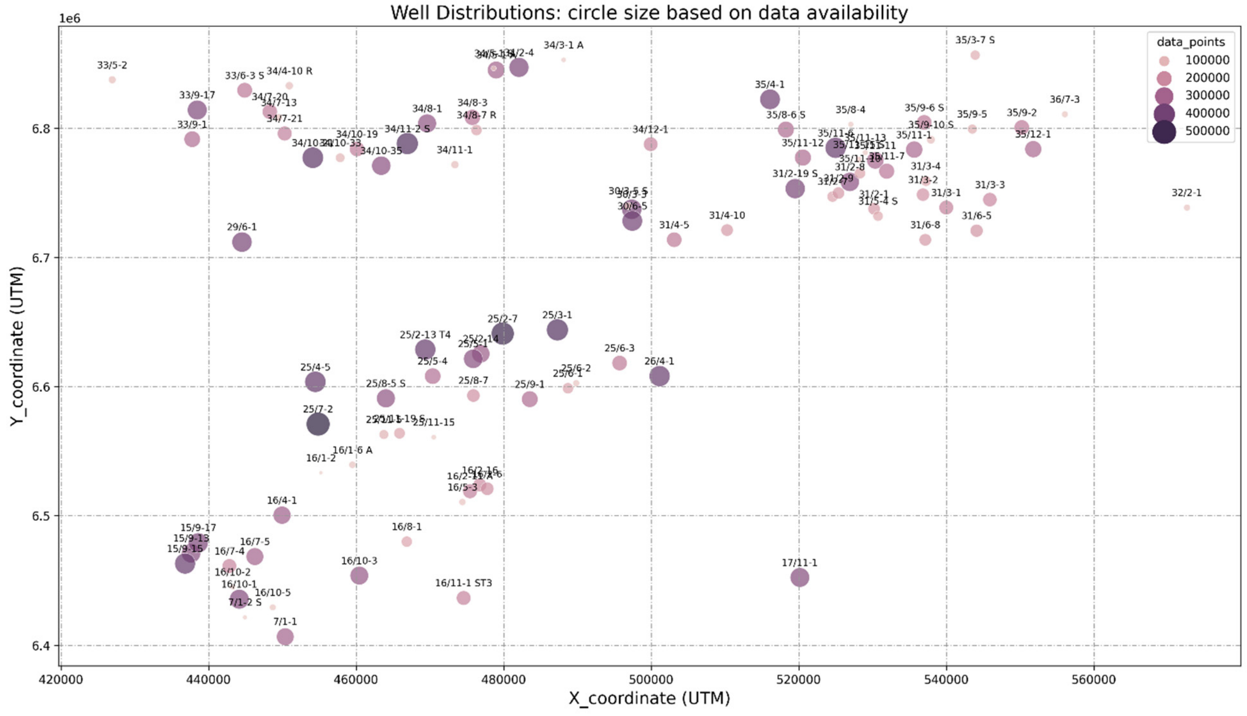
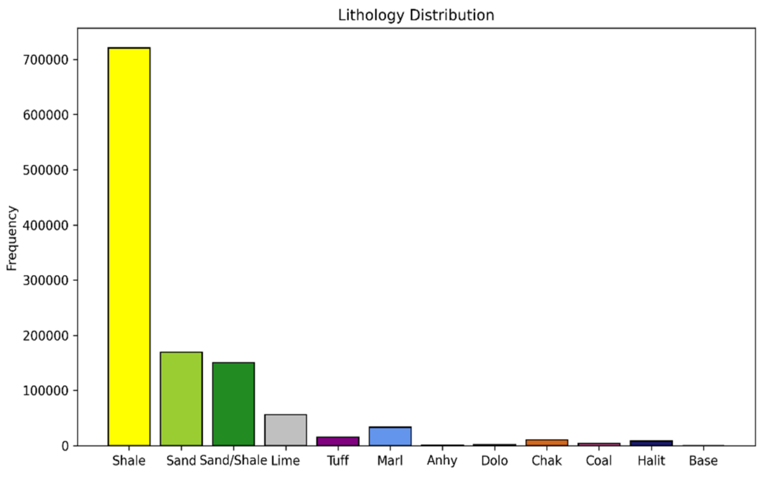


FIG. 1. Well distribution map. Circle size is related to available data in that well.



Class	Count	Percent
Shale	720803	61.58
Sand	168937	14.43
Sand/Shale	150455	12.85
Lime	56320	4.81
Tuff	15245	1.30
Marl	33329	2.85
Anhy	1085	0.09
Dolo	1688	0.14
Chak	10513	0.90
Coal	3820	0.33
Halit	8213	0.70
Base	103	0.01
Sum	1170511	1

FIG. 2. Lithology distribution shows shale is the major class, and the other classes are either moderately or extremely imbalanced.

Figure3 shows the most important and common logs that we can see in this dataset. From the left, measure depth (meters), formation tops, bit size and calliper, GR and SP, resistivity logs, neutron and density, PEF and sonic slowness, mud weight and rate of penetration, interpreted lithology, and interpretation confidence is depicted. Statistically, lithologies are nominal categorical values while almost the other columns are continuous numerical variables except formation tops (similar to lithology).

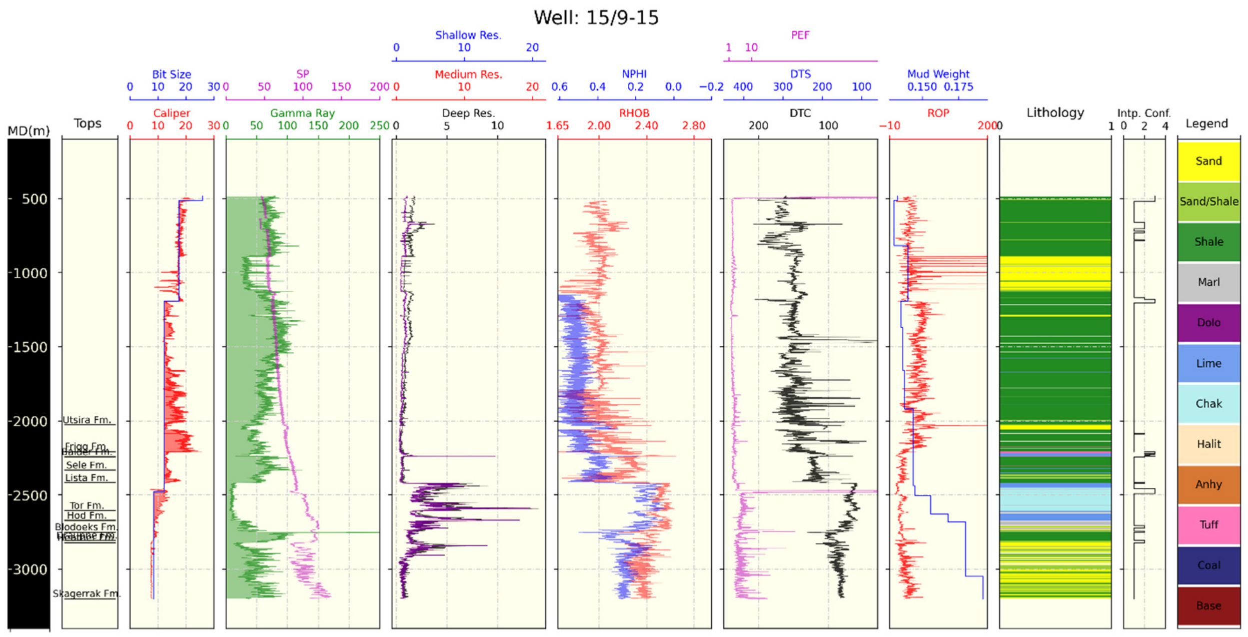


FIG. 3. Plot of log and lithology for a single well.

METHODOLOGY

We used the python programming platform and required libraries/packages in this research. Pandas is used for data analytics and preparation, Matplotlib and Seaborn for plotting, Imbalanced-learn for balancing and scikit-learn, and XGBoost for training algorithms.

Data Pipeline

Real-world datasets are not clean enough to be consumable by any machine learning algorithms. There are several standard steps for data preparation that should be done in sequence. For this project, we defined a custom data pipeline that includes all required steps as shown in Figure 4.

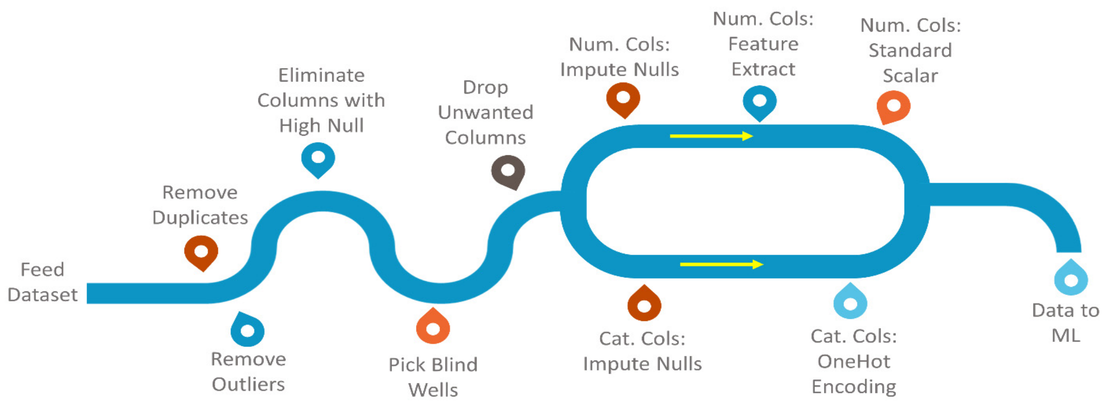


FIG. 4. Our data pipeline consumes original raw data and exports data ready for ML algorithms in tabular format.

Our data pipeline implements these steps:

- Remove duplications: duplicates can mislead training algorithms.

- Remove outliers: outliers can cause the model to learn incorrect data as they can have a huge impact on feature mean values. The data point is supposed to be an outlier if it is over 3 standard deviations from the mean value.

- Eliminate columns with high null values: some columns (logs) have a lot of missing values. During running the data pipeline, you can specify a threshold for missing data to be preserved for the training process. We used 70% for this research meaning if 70% of the samples in a single column are missing, that column will be removed from feature vectors.

- Pick Blind Wells: after training and validating any model we need to examine the model's performance. To make it happen we can pick up some wells as blind well that will not expose to the training process. We need to carefully select wells as it is important to have all class samples present in the blind wells.

- Drop unwanted columns: some columns should be dropped as they don’t have related information like well name or cause a co-linearity problem like measure depth and total vertical depth.

- In the next step, categorical and numerical columns will experience different processes:

- Impute numerical null values: models can not handle missing data. To impute missing values, we can use mean or median values.

- Feature extraction: sometimes to enhance model performance, we can calculate some features from existing feature vectors. We calculated total porosity, acoustic impedance, shale volume, and sonic porosity from existing log data using a custom transform class in python.

Total Porosity: $$PHIT = \sqrt[2]{\frac{RHOB^2+NPHI^2}{2}} \quad (1)$$

Where RHOB is bulk density and NPHI is neutron porosity.

Acoustic Impedance: $$AC_{IMP} = \frac{1e6}{DTC*RHOB} \quad (2)$$

Where DTC is sonic log slowness and RHOB is bulk density.

Shale Volume: $$Vsh = \frac{GR - GR_{min}}{GR_{max} - GR_{min}} \quad (3)$$

Where GR is the Gamma Ray reading.

Sonic Porosity: $$SPHI = \frac{DTC - DTC_{ma}}{DTC_{fl} - DTC_{ma}} \quad (4)$$

Where DTC is sonic log slowness and DTC_{ma} is matrix slowness and DTC_{fl} is fluid slowness (salt water).

- Standardization: some ML algorithms are sensitive to feature ranges, so it is better to normalize or standardize the data range. This process can speed up computation as well.

- Impute categorical null values: some categorical columns also have null values. We can use mean, median, or adjacent row values to impute null values.

- One-hot encoding: ML algorithms can not handle categorical values directly. So, we need to encode categorical values in a dataset like formation tops into numeric values.

Sampling

There are two sampling methods available to handle the imbalanced data:

Under Sampling: this technique helps balance data by eliminating some samples from major classes. This method should be used with caution as reducing skewness can lead to information loss during the training process. We will not use this method for this project.

Over-Sampling: unlike the under-sampling method that focuses on decreasing major classes, over-sampling increases minority class samples. In this work, we will use different methods of over-sampling to handle imbalanced data.

We need to emphasize that although the over-sampling technique balances the data but does not add new/additional information to the dataset. It increases the chance that minority groups can be seen by the model more than whenever it is imbalanced.

We plotted (Figure 5) a very small random sample of the dataset (200 samples) in the space of GR and RHOB but for simplicity, we selected three facies: shale (purple), chalk (green), and halite (yellow) with 181, 13 and 6 samples, respectively.

Random Over Sampling:

This is the most naive strategy to generate samples by randomly sampling with replacement of the currently available samples (Figure 5, upright). Using this method, we add some repetition to minor classes. If repeating samples is an issue in the training process for ML algorithms, which is for some algorithms like decision trees, we can perturb the original samples to generate more smoothed results.

Synthetic Minority Oversampling Technique (SMOTE):

In this technique, the operator selects the nearest sample in the feature space, then draws a line between them and creates a sample on this line (Figure 5, down left).

"SMOTE first selects a minority class instance at random and finds its k nearest minority class neighbours. The synthetic instance is then created by choosing one of the k nearest neighbours b at random and connecting a and b to form a line segment in the feature space. The synthetic instances are generated as a convex combination of the two chosen instances a and b" (He and Ma, 2013).

Adaptive Synthetic Sampling (ADASYN):

This technique works based on the density of minority class instances. This method is similar to SMOTE, but it generates a different number of samples depending on an estimate of the local distribution of the class to be oversampled (Scikit-learn, 2011).

Both SMOTE and ADASYN employ the same algorithm to generate new samples. Considering s sample X_i, a new sample X_{new} will be generated using its k nearest-neighbours (Figure 6). For example, if the k is 3, one of these points, X_{zi} will be selected, and the sample generated using this equation:

$$X_{new} = X_i + \lambda * (X_{zi} - X_i) \tag{5}$$

where λ is a random number in the range [0,1]. This interpolation will create a sample on the line between X_i and X_{zi} as illustrated in Figure 6 (Scikit-learn, 2011).

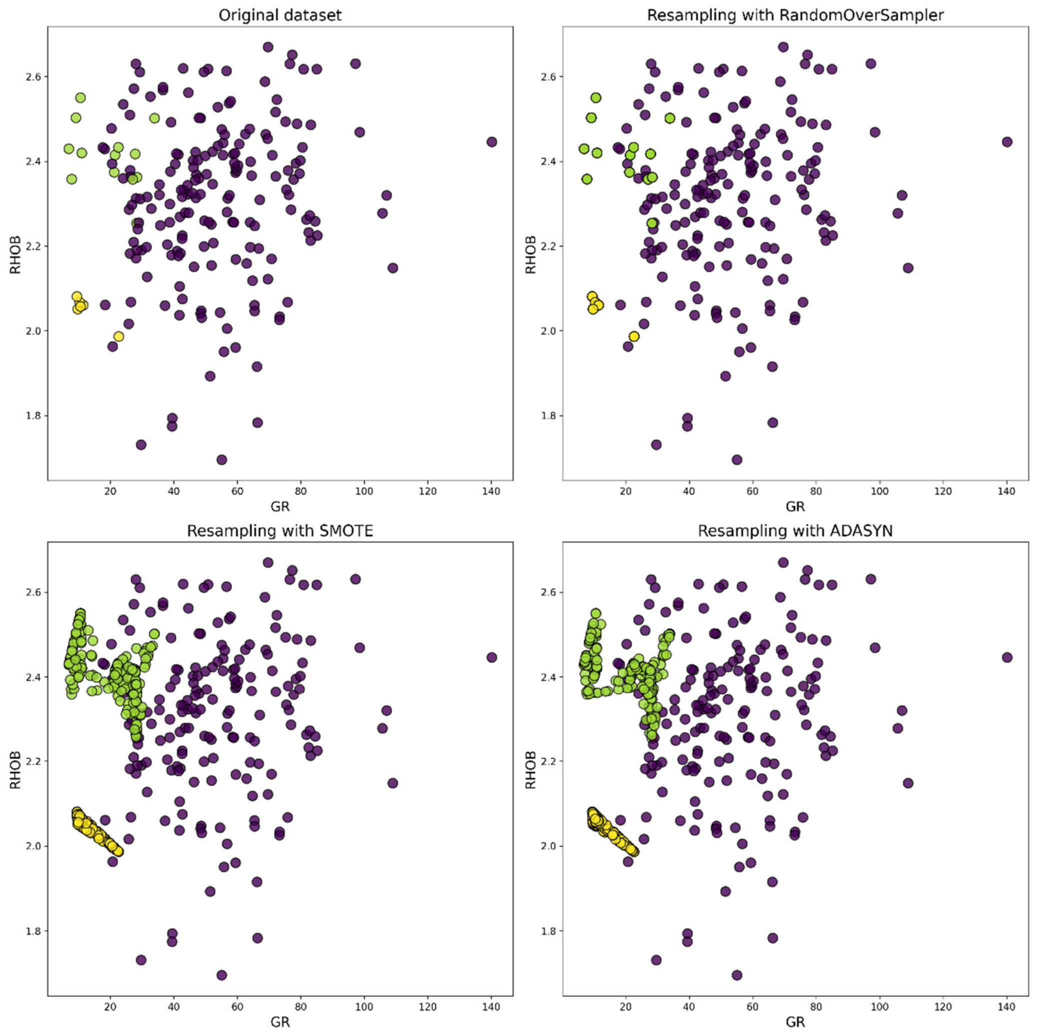


FIG. 5. Small random sample of a dataset with 200 samples plotted on GR and RHOB dimensions. From 12 lithology classes, we selected three of them as shale (purple), chalk (green), and halite (yellow). Shale is the major class and the others are moderately and extremely imbalanced samples, respectively. The random over-sampling method added samples randomly though without modifying data variance (top right). SMOTE and ADASYN resampling result is plotted in the second row. Both act like yellow samples while small differences can be seen in green data points resampling.

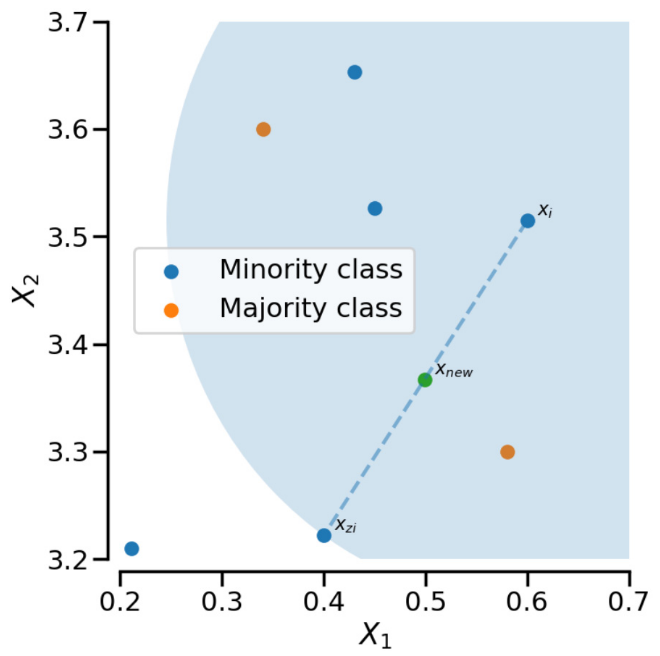


FIG. 6. from X_i to X_{zi} new sample will be generated considering a random coefficient (from, imbalanced-learn.org).

After examining different over-sampling methods and visualization on a small random sample of the dataset, we ran those methods on the whole dataset. The training sample increased from 1,035,963 rows to 6,795,996 instances. Features stay the same as it is 21. This is also an important point that we shouldn't resample testing datasets.

We selected the XGBoost training algorithm for this research because of two main reasons: first, this algorithm is the FORCE 2020 contest winner for this dataset, and we know the exact hyperparameter values of this algorithm. Second, we can run this ML model in GPU to get the benefit from the parallel computation. XGBoost is boosting algorithm that works sequentially by adding predictors to an ensemble, each one correcting its predecessors. This method fit the new predictors on the residual errors which resulted from the previous predictor (Geron, 2019).

RESULTS AND DISCUSSION

Before resampling the dataset (after picking blind wells) we divided it into training and validation sets with a ratio of 9:1. Training dataset was resampled using different techniques and examined model performances on the validation dataset. After implementing model evaluation using classification metrics for imbalanced data, we examined model performances on blind wells (testing dataset). In practice, we are going to have four gradient-boosting model results:

1- A model trained on the original dataset (not resampled)

2- A model trained on the dataset resampled using the Random Over Sampling technique

3- A model trained on a dataset resampled using the SMOTE technique

4- A model trained on a dataset resampled using the ADASYN technique

Let's start model evaluation first by comparing the true facies counts in blind wells and model prediction results. Figure 7 shows real and predicted facies distribution for each class.

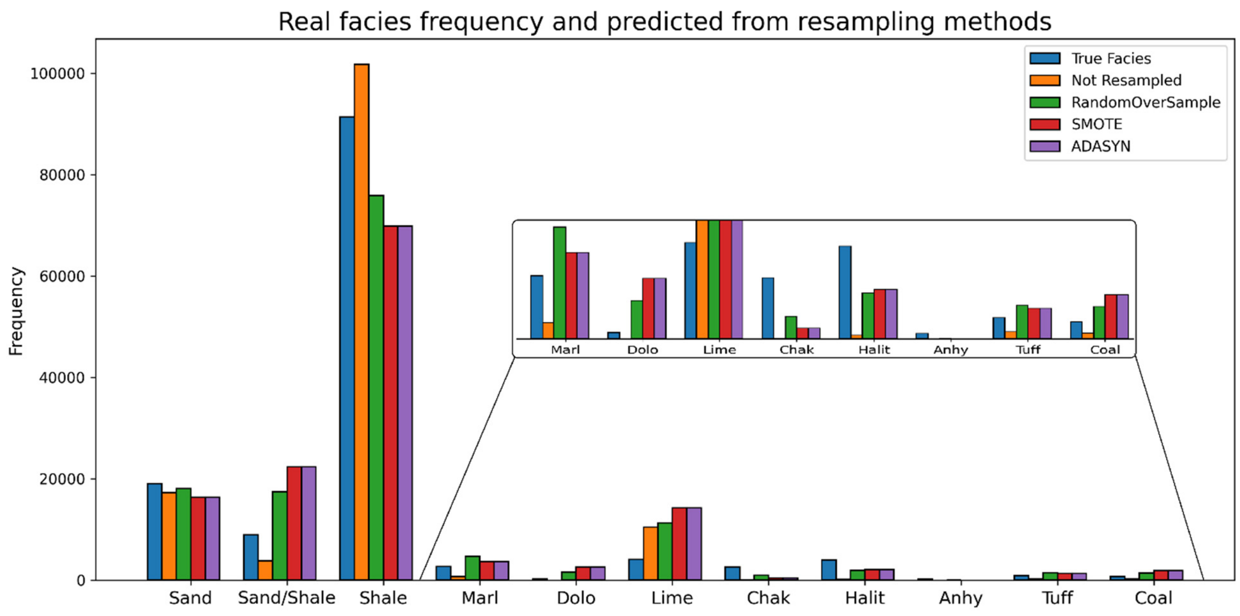


FIG. 7. The frequency of real and predicted facies classes is plotted. Overall, resampling improved the prediction of minorities.

We can see these trends:

- Balancing caused predicted facies counts to double or more for some classes like Sand/Shale, Limestone, and Dolostone.
- XGBoost algorithm over-predicted the majority class (shale) when feeding the model with the original (none-resampled) training dataset (orange bar in Shale class). This also happened to Lime facies.
- Zooming to extremely imbalanced classes reveals that the model could not perform well enough if we don't implement balancing (the orange bar is almost not present in minority classes except limestone)
- Balancing caused minorities over-predicted in most cases. Dolomite predicted almost three times more than actual counts.

- Over-sampling didn't improve Anhydrite prediction

Let's dive into common evaluation metrics.

Precision

This metric tries to answer this question: what proportion of positive identifications was actually correct? So, it is defined as:

$$\text{Precision} = TP/(TP+FP) \quad (6)$$

Where TP is a true positive, FP is a false positive.

Let's look at Figure 8. The prediction result is plotted for the four modelling scenarios for all facies classes. The first group of bars is for sand, and it has about 80% precision. This means that 80 percent of those predicted sand class are truly sands. 20% are non-sand (can be any of the other lithologies). Marl and Dolostone show very weak precisions. Resampling increased the chalk identification actual rate, noticeably. Random over-sampling is the one that could help Anhydrite precision jump to 100%. Remember, you can easily get 100% of precision by sacrificing some true members not to be involved in the prediction, minimizing false positives. This concept leads us to the necessity of recall.

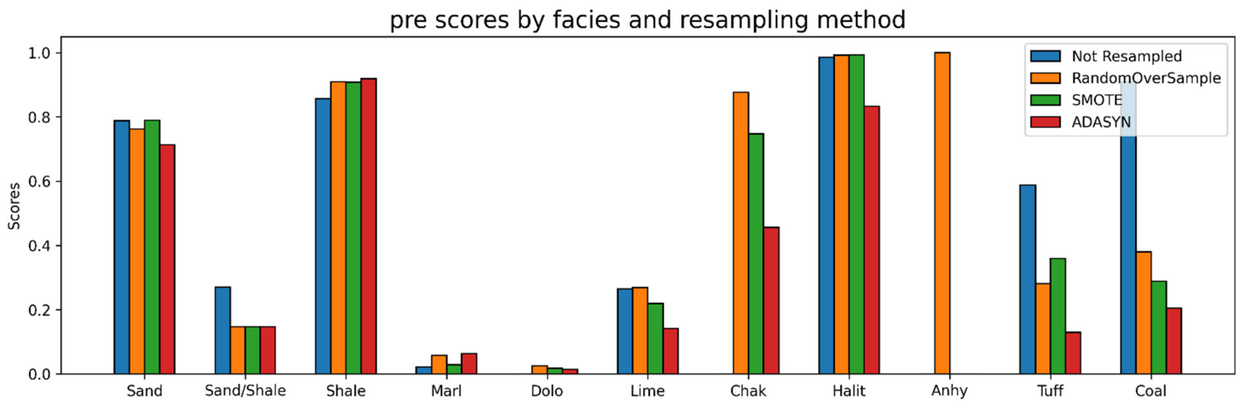


FIG. 8. Precision scores for different facies predictions.

Recall

This metric tries to answer this question: What proportion of actual positives was identified correctly? So, it is defined as:

$$\text{Recall} = TP/(TP+FN) \quad (7)$$

Figure 9 shows the bar chart of recall for predicted classes. This metric says that actual members of a class like sand are recognized correctly by the model. In another word, 75% of actual sand samples in our dataset is correctly predicted by the none-resampled XGBoost algorithm. Here, we notice that Sand/Shale, Tuff, and Coal recall doubled by the balancing methods. Anhydrite's actual samples were not correctly classified by none of the models. Random Over sampling and SMOTE improved halite's recall noticeably.

Unfortunately, there is a negative correlation between precision and recall. Improvement in one will lead drop in the other one. We can set a threshold on the model to stop training for specific precision/recall. This trade-off can be decided by business objectives. There is another metric that is a combination of these two metrics called the f1-score.

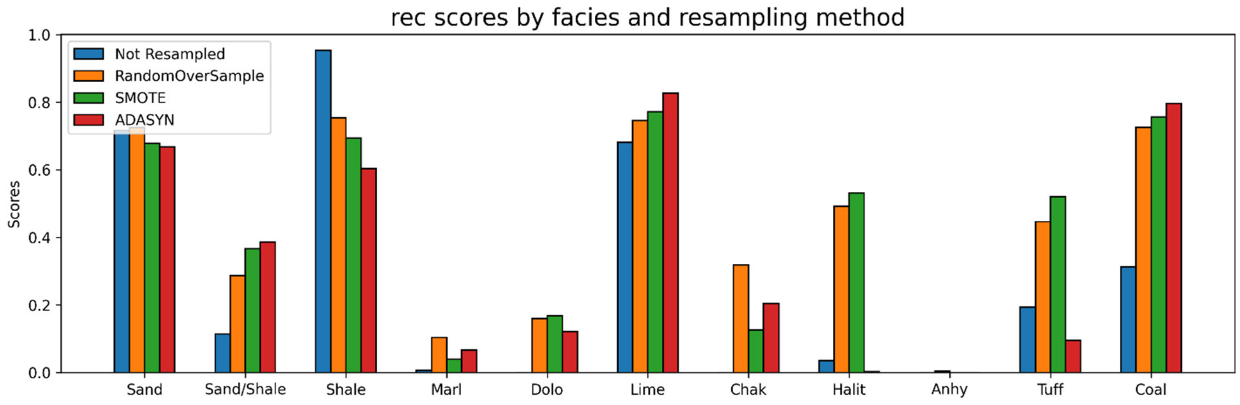


FIG. 9. Precision scores for different facies predictions.

F1 Score

To compare different classifiers' prediction qualities, we can use the F1 score which is the combination of precision and recall. It is the harmonic mean of precision and recall. While the regular mean treats all values equally, the harmonic mean gives more weight to low values. As a result, the classifier will have a high F1-score if both recall and precision are high. It is defined as:

$$F_1 = 2\frac{precision*recall}{precision+recall} = \frac{2TP}{2TP + FP + FN} \tag{8}$$

Let's look at Figure10. For the majority class, we have a drop in model performances when balancing happens. F1 is improved for marl and dolostone when we balanced the dataset but still noticeably low. Except for coal, the other minor classes' F1 scores are improved if we used oversampling techniques. Anhydrite, as expected from low recall, didn't receive an acceptable f1-score. Overall, SMOTE performed well enough to be the candidate for all minor class's resampling techniques.

In the last step of model performance and prediction results based on balancing methods, let's plot the predicted facies along the true classes in the one well (Figure 11). In the first five tracks of the logs, we plotted features like GR and NPHI, and so on. Other tracks are the predicted facies. Track six is the XGBoost's prediction result on the original dataset (not balanced). The other tracks are the prediction result on the dataset balanced by

Random Over Sampling, SOMTE, and ADASYN. True lithology is plotted at the last facies track.

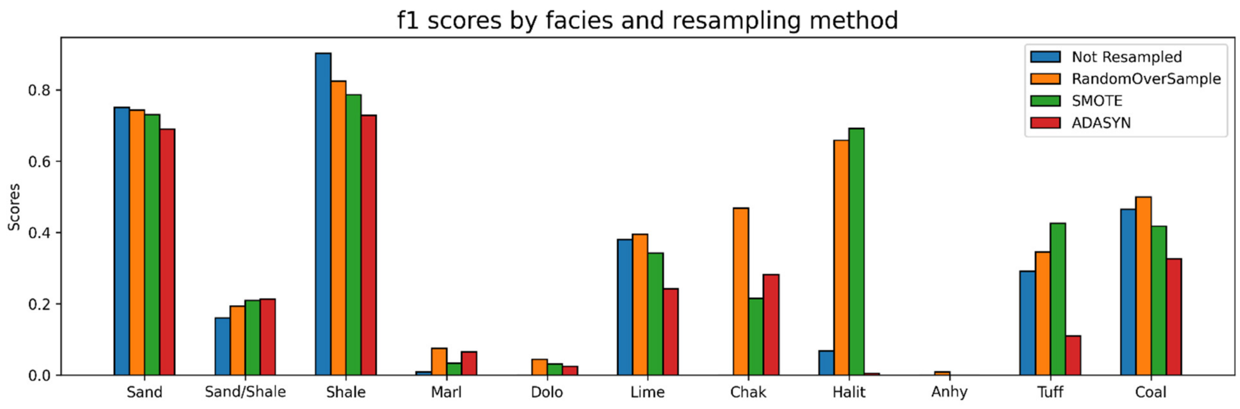


FIG. 10. F1 scores for different facies predictions.

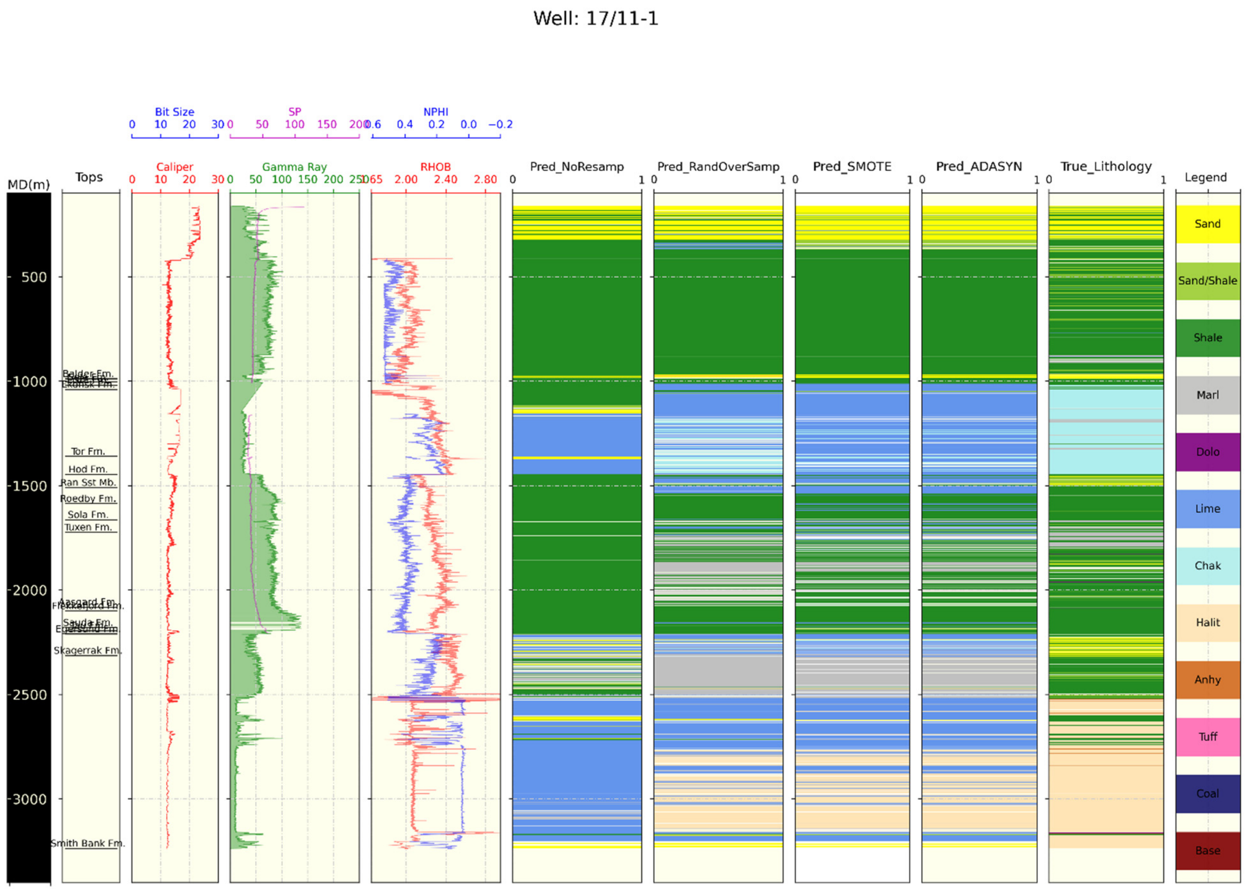


FIG. 11. Plot of some well logs (predictors) and interpreted facies using various resampling methods along with actual lithology (well: 17/11-1).

There are some clear points in this plot:

- Chalk's prediction was not successful. Almost all balancing methods caused chalk to be predicted as limestone except for some intervals by random over-sampling (depth 1000m-1500m). If we back to Figure 7 we can see that predicted lime counts tripled the original frequency.
- Dolomite couldn't be predicted by any of the resampling methods
- The thick layer of halite in the deep intervals could be predicted by all balanced methods. The model prediction on an imbalanced dataset predicted halite as limestone

Let's look at another well (Figure 12). The main key points in this plot are:

- Major classes are predicted perfectly
- SMOTE and ADASYN predicted shales at shallow intervals as limestone
- Tuff and coal could be detected successfully by SMOTE and ADASYN resampling methods

As you may notice, some minor classes could be predicted better by oversampling techniques, especially SMOTE did an effective job. Some other classes, on the other hand, have been over-predicted like limestone. Here the concept of business problem importance comes to play. Suppose Tuff is a very important facies and when you drill a well, you don't want to miss that interval. In such cases, we prescribe running oversampling techniques on the imbalanced dataset before feeding it to intelligent systems. Or suppose marl is a dangerous zone for drilling and there is a chance of drilling bit stuck in such intervals. In such case, we don't want to miss any marl interval (high recall) though some other rock could be predicted as marl and we need to screen the model result manually. This will cost the operator to supervise the drilling operation to manually help the model by live lithology interpretation during operation.

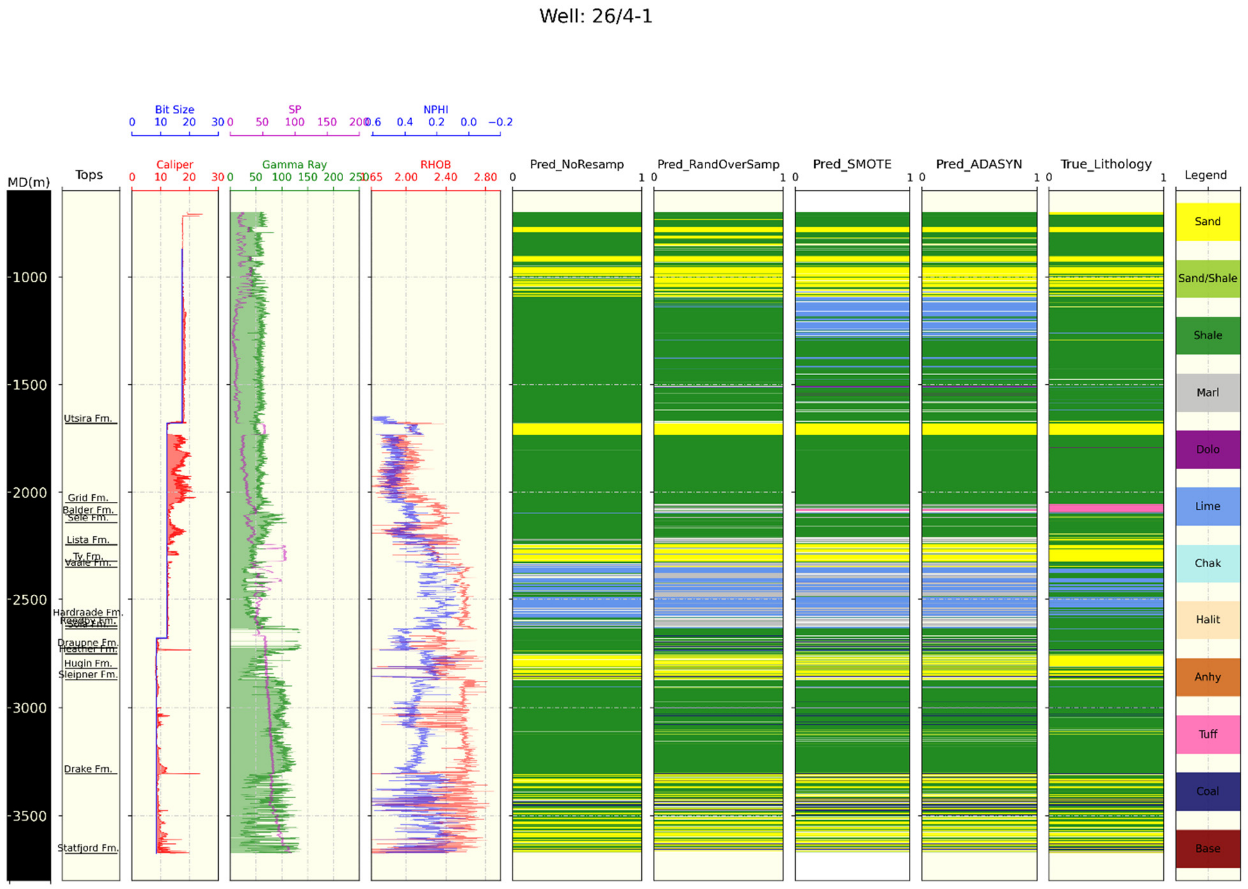


FIG. 12. Plot of some well logs (predictors) and interpreted facies using various resampling methods along with actual lithology (well: 26/4-1).

SUMMARY

This was a multiclass classification problem in that we employed the XGBoost algorithm to predict lithology from well logs. The objective of this research was to examine how oversampling methods can affect the prediction result on the imbalanced dataset. We observed that oversampling techniques could help the classifier to predict minority classes with higher accuracy than when it is imbalanced. Overall, SMOTE is the better candidate for oversampling, though for some classes ADASYN could beat the SMOTE performance.

REFERENCES

Geron, A., 2019, Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: concepts, tools, and techniques to build intelligent systems (2nd ed.). O'Reilly.

He, H. and Ma, Y., 2013, Imbalanced Learning: Foundations, Algorithms, and Applications. Wiley-IEEE Press, Hoboken, NJ, USA.

Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.

https://imbalanced-learn.org/stable/auto_examples/over-sampling/plot_illustration_generation_sample.html