Data augmentation based on the Wasserstein GAN with gradient penalty to enhance the prediction of genes associated with RNA methylation pathways

**Abstract:**

RNA methylation modification influences various processes in the human body, which has gained increasing attention from scholars. Predicting gene associated with RNA methylation pathways can significantly aid biologists in their study of RNA methylation processes. Limited prediction methods have been investigated; their performance is still constrained by the limited sample amount. To improve the prediction of genes associated with RNA methylation pathways, we utilized the dataset collected from article [4] and implemented deep learning techniques combined with five well-established machine learning classifiers to oversample the minority data. The experimental results indicate that the synthetic gene samples can complement the original sample distribution effectively, resulting in an improved prediction performance for genes associated with the RNA methylation pathway.

整篇文章注意以下这两个准则

1. 描述你主要的方法（方案），对于其它方案只是用来证明你采用的这个方案好（所以不应该把SMOTE和WGAN without block对比）

2. 实验部分只要保留实验的设置，实验结果的描述，以及实验结果的分析。所有你所采用的方案放到方法中

1. **Introduction:**

Introduction章节各段参照如下步骤：

1. 介绍RNA甲基化（参考communications biology的第一二段）

2. RNA甲基化很重要，但是我们知道的信息依然很少（参考第三段）

3. web-lab的局限性，计算方法的优势，以及现有计算方法的劣势，数据增强方法有望提升性能（参考第四段）

4. 介绍一些data augmentation方法（参考https://www.sciencedirect.com/science/article/pii/S0957417420305200?casa\_token=bMkNt5l0ykcAAAAA:JlMikYRHQWJl04eBfByZhG-3s86igtvZhcEk\_s\_X7Q2S-F\_ufV7uRl6ePIML43LGBRIEkyR5Rw的introduction的倒数第二三段）

5. 自己的方法及实验介绍

Researchers have discovered over 160 RNA modifications [5] that are extensively distributed across eukaryotes and prokaryotes. Among them, the most prevalent form of RNA modification is methylation. RNA methylation refers to the chemical process in which methyl groups are selectively added to methyladenine in RNA under the catalytic action of methyltransferase. Through studying this process, researchers have discovered associations between RNA methylation modifications and various fields, including cancer, cardiovascular disease, embryonic development, and cell differentiation. These findings highlight the significance of RNA modifications and suggest their potential implications in diverse biological processes and diseases. Further research in this area may shed more light on the functional roles of RNA modifications and their therapeutic applications.

Numerous reports have revealed that specific enzymes or proteins within the RNA methylation pathway exert regulatory effects on distinct functions and biochemical processes in the human body. These studies are founded on the identification of proteins, genes, or specific enzymes associated with RNA methylation. Despite the significant role of the RNA methylation process in various aspects of the human body, researchers have limited knowledge about the pathways involved. This limitation primarily arises from the substantial financial and time investments needed to identify gene functions using experimental wet laboratory methods. Fortunately, due to intensive research of large-scale omics data and artificial intelligence methods in this field, it is now possible to use computational methods that are faster and more cost-effective than wet methods. Researchers can explore the function of target genes through various computational methods.

Several reports have explored the classification of RNA methylation pathways genes. Researchers collect data and conduct experiments using established machine learning models on known RNA methylation pathways genes as positive samples. This suggests that bioinformatics methods may be used to identify RNA methylation pathways genes. However, due to the limited amount of known samples and the imbalanced ratio of positive and negative samples, avoiding overfitting and improving the classification method's effectiveness remains challenging. Therefore, to address this sample imbalance issue, we oversampled known positive samples (i.e., RNA methylation pathways genes) and conducted dry experiments based on other successful deep learning algorithms. Our aim was to reduce overfitting and improve the classification performance for identifying gene functions.

**2. Method**:

**2.1. Workflow of the study**

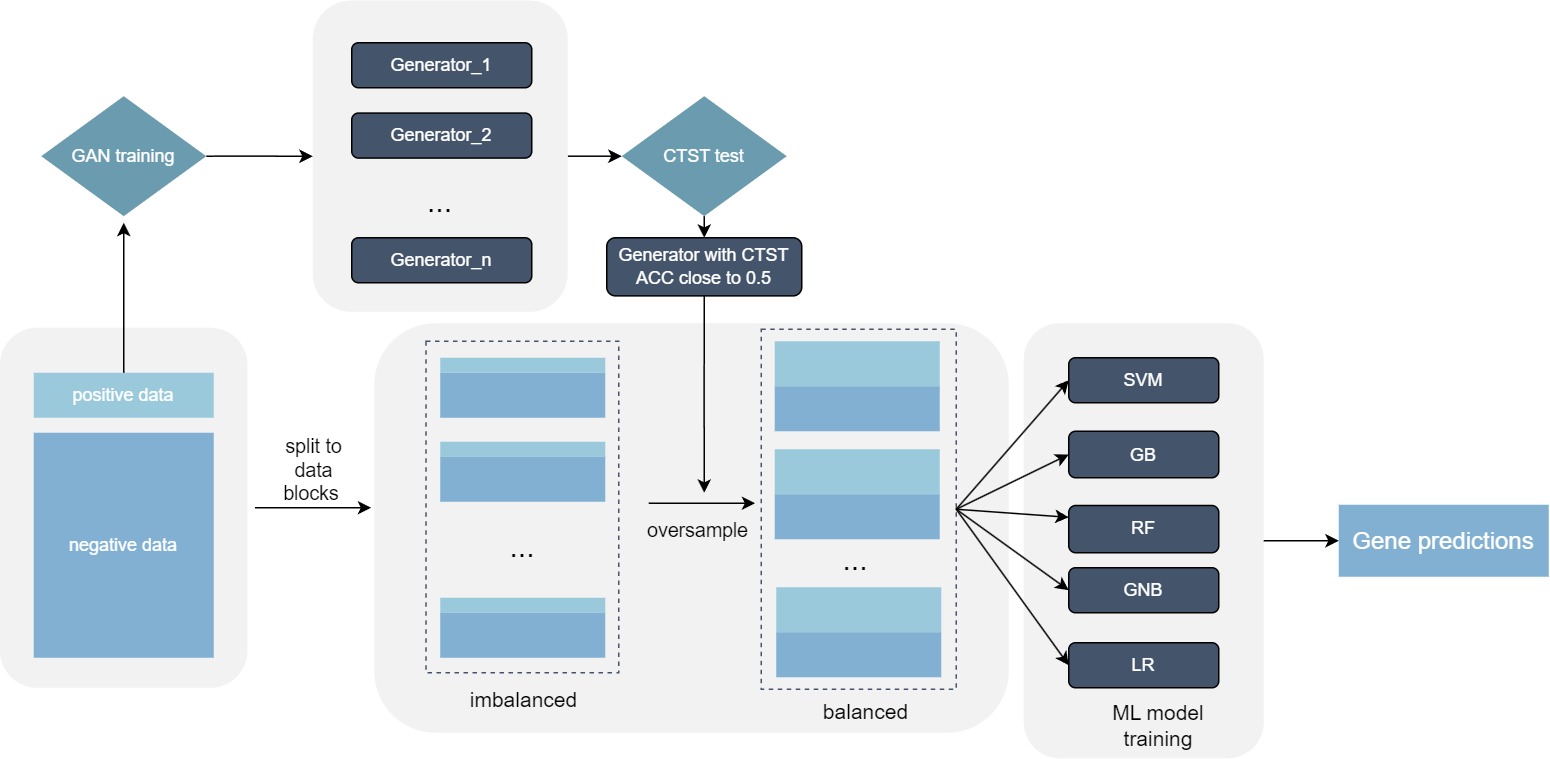
Generally, our proposed method consists of three steps aimed at improving the prediction of genes related to the RNA methylation pathway. In the first step, we use known genes associated with RNA methylation (referred to as positive samples) as the training set. Next, we adopt a WGAN with gradient penalty to fit the distribution of positive samples and obtain generators for oversampling in downstream tasks. In the second step, we evaluate all saved generative networks using Classifier Two-Sample Test (CTST) and select the network with the best accuracy(close to 0.5) to generate synthetic positive samples for downstream tasks. Finally, we utilize the network selected in the second step to oversample during the training phase of five different machine learning classifiers, resulting in improved classifier performance and higher prediction accuracy.

Figure 1: Workflow of our study

A screenshot of a graph

Description automatically generated（用一个这样图表示classifier）

**2.2. Data set and feature transformation**

We use the dataset created and made available by Tsagkogeorga *et al*. Original dataset collected from Harmonizome website(<https://maayanlab.cloud/Harmonizome>). 15 one-hot-encoded datasets were selected to construct our dataset. The dataset was initially standardized to continuous-value ranged from 0 to 1, or -1 to 1 where 1 indicated strong positive gene-feature association, 0 indicates no gene-feature association observed and -1 indicates strong negative gene-feature association.

The genes within these 15 datasets, along with their associated features, exhibit variations. The quantity of genes and features differs across datasets. Certain genes can be present in one or multiple datasets, while specific features may be present across various datasets as well. Utilizing pandas to merge the data based on the gene's official symbol results in a dataset comprising 26,936 genes. Each gene encompasses 50,176 features, forming a matrix of dimensions 26,936x50,176. Within the 15 datasets, certain genes lack specific features that are prevalent in other datasets. Consequently, features absent from a gene in this context will be marked as null. Subsequently, the values within the dataset marked as null will be assigned a value of 0. While the presence of 0 within the dataset signifies the absence of gene-feature associations, this will not impact the ultimate outcomes of training and prediction.

To gain high-informative features and reduce feature dimensions, following 2 kinds of features will be removed: (i) zero values more than 70%. (ii)variance less than 16% of whole.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Train & CV | Test | Total |
| Positive gene numbers | 74 | 18 | 92 |
| Negative gene numbers | 21044 | 5260 | 26304 |
| Total | 21118 | 5278 | 26396 |

Table 1: Numbers of gene sample for training, CV and test

**2.3. The small-sample augmentation based on GAN**

**参照<https://ieeexplore.ieee.org/abstract/document/9130768?casa_token=3dEbyEzWsUwAAAAA:VT21szdjU8O3kSvVRINQl9v_EBz4N7glmPI5zuA3DmzoXE3hw9x0qqfECDeNiCDFcm1zl2Na>**

**https://www.sciencedirect.com/science/article/pii/S0957417420305200?casa\_token=bMkNt5l0ykcAAAAA:JlMikYRHQWJl04eBfByZhG-3s86igtvZhcEk\_s\_X7Q2S-F\_ufV7uRl6ePIML43LGBRIEkyR5Rw**

**这两篇文章描述你的WGAN**

WGAN-GP, an acronym for Wasserstein Generative Adversarial Network with Gradient Penalty, represents an improved iteration of Wasserstein GAN. Initially introduced in 2017 by Martin Arjovsky, Soumith Chintala, and Léon Bottou, WGAN-GP's fundamental concept extends from Wasserstein GAN and substitutes the weight clipping of the original WGAN with a gradient penalty integrated into the loss function. This modification tackles potential problems associated with weight clipping.

By incorporating this gradient penalty technique, the training of the discriminator gains stability, circumventing potential drawbacks associated with weight clipping. Precisely, this penalty enforces gradual adjustments in the discriminator's output throughout the data distribution encompassing real and generated samples, mitigating excessively sharp gradient transitions. The intensity of this penalty is regulated by a parameter referred to as the penalty coefficient, requiring calibration during training to achieve a performance equilibrium between the generator and discriminator.

WGAN-GP's loss function consists of two elements: one related to Wasserstein distance and the other associated with the gradient penalty. Collectively, these constituents constitute the discriminator's loss function. Through the incorporation of gradient penalty, WGAN-GP adeptly addresses potential challenges inherent in conventional WGAN configurations. This achievement positions it as a notable progression in the realm of generative adversarial networks.

We trained a WGAN-GP to match the distribution of positive samples in the training set. The GAN's generator was used to oversample the minority set. To achieve the optimal fit for the distribution of positive samples, 73 positive samples were used during GAN training. Recognizing the instability of GAN training, we introduced the early-stopping parameter STOPTHRESHOLD. We used SVM to conduct a classifier two-sample test (CTST) at the end of each epoch. The accuracy of SVM's cross-validation served as the outcome of CTST. Models with a difference from 0.5, as indicated by CTST results, lower than the STOPTHRESHOLD, were saved.

After training the generator, we performed an additional CTST on all the retained generators. Each generator produced synthetic samples equivalent to 20% of the positive sample count, and the CTST experiment was conducted using SVM. CTST was repeated multiple times on each generator. Generators with an average CTST result closest to 0.5 were chosen and employed for a similar oversampling training approach as SMOTE.

**2.4. Prediction models**

**是否可以用什么deep learning model能得到不错的结果，MLP的结果如何？（注意调参）**

**参考R-WGAN-Based Multitimescale Enhancement Method for Predicting f-CaO Cement Clinker的”D. Regression Prediction Networks (R)”**

**参考SSP-WGAN-Based Data Enhancement and Prediction Method for Cement Clinker f-CaO的“C. Prediction Network”**

**上面这篇文章的预测部分用的简单的DNN或者拼两个，你用类似的方法做classifier**

**2.4. Training and evaluation**

We employed five machine-learning classifiers: SVM, GB, GNB, RF, and LR. These classifiers were trained on balanced data blocks. To select the best hyperparameters for SVM, GB, and RF, we conducted grid-search using a 3-fold cross-validation approach. None-sample method (OSRATE = 1) was set as baseline.

For our GAN model, we opted for WGAN-GP and conducted multiple repetitions of CTST to identify the optimal generator. The parameters for SVM in CTST were kept at their default values, and the STOPTHRESHOLD was set to 0.02. To tailor the GAN for gene expression data, we enhanced its architecture by replacing CNNs with fully-connected layers in both the generator and discriminator. Each fully-connected layer was followed by a leaky ReLU activation function (with a negative slope of 0.2).

The discriminator comprised four fully connected layers, with an input dimension of 1517. The output size of each layer was half that of its input, culminating in a final output size of 1. For the generator, with an input size of 128, each layer's output size was double that of its input, resulting in a final output size of 1517, matching the data dimension. We optimized the GAN using the ADAM optimizer with parameters beta1 and beta2 set to 0.95 and 0.9, respectively. The batch size was 73, and the learning rate for both discriminator and generator was set at 0.0001.

We introduced the parameter NCRITIC, set to 7, signifying that the generator underwent training once after every 7 training iterations of the discriminator. Initially, we set the number of epochs to 1500, but the generator effectively captured the training sample distribution after approximately 600 to 800 epochs. Finally, the parameter OSRATE in both SMOTE and GAN methods was set to 0.5, resulting in half of the positive training samples being generated by either SMOTE or GAN.

**3. Result:**

**3.1. Influence of hyperparameters and training strategies**

To explore the impact of oversampling rate on model performance, we evaluated the model performance for different oversampling rates. The results showed that the model performance was affected differently under different oversampling rates. The two oversampling methods perform best at an oversampling rate of 0.6. As the number of oversampling increases (the oversampling rate decreases), the number of negative samples in each data block relatively increases, and the effectiveness of the oversampling method decreases. Specifically, the SMOTE method experienced overfitting, with a significant decrease in the average levels of recall and F1 SCORE at an oversampling rate of 0.2. On the contrary, the WGANGP method showed a slight decrease in performance and overfitting at an oversampling rate of 0.2, but the situation was much better than SMOTE. We speculate that this phenomenon may be due to an increase in negative samples, while the samples produced by SMOTE simply take a point in the middle of the Euclidean distance of the feature space as the feature value of the sample, resulting in severe overfitting. This also indicates from another perspective that the samples generated by the WGANGP method have better diversity and are more in line with the distribution of the original positive samples.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 0.2 | 0.4 | 0.6 | 0.8 |
| SMOTE | 26.106 | 75.645 | 82.608 | 82.909 |
| WGAN-GP | 84.646 | 85.380 | 84.618 | 83.283 |

Table 2: MCC test result among different oversampleing rates, based on SVM

Given the significant imbalance between positive and negative samples in our dataset, directly applying oversampling may not enhance classifier performance. Therefore, we further adopted a dataset block constructure for oversampling. This approach combines the advantages of undersampling and oversampling techniques, mitigating information loss and overfitting to some extent. Additionally, we performed oversampling multiple times on each data block instead of directly oversampling and balancing the entire dataset. This allows the classifier to better learn the boundaries between positive and negative samples.

3.2 ablation study

To examine the impact of data blocks on classifier performance, we conducted ablation experiments excluding the use of data blocks. Specifically, the original dataset comprises 26,304 negative samples and 92 positive samples. After allocating 80% of the samples to the training set, there are still 21,044 negative samples and 74 positive samples. We employed SMOTE and trained generators to generate 20,907 synthetic positive samples based on the original data, ensuring a balanced positive-to-negative sample ratio for the entire dataset. Subsequently, we employed the same method for classifier training and testing.

It is important to note that since the training samples are not trained in blocks, each classifier will ultimately only receive one model file. To maintain experimental control over variables, we continued to employ the block testing mode during testing while maintaining consistent scoring standards.

Overall, our approach of using the dataset block pattern for oversampling addresses the severe imbalance issue and leverages the benefits of undersampling and oversampling techniques. By iteratively oversampling each data block and ensuring a balanced distribution within each block, we enable the classifier to better discern positive and negative boundaries. The ablation experiments and consistent testing methods further enhance the reliability and comparability of our results.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Accuracy | Precision | Recall | F1 |
| SVM | 0.57885 | 0.99911 | 0.15789 | 0.27269 |
| RF | 0.68421 | 1.0 | 0.36842 | 0.53846 |
| GB | 0.68421 | 1.0 | 0.36842 | 0.53846 |
| GNB | 0.87841 | 0.87091 | 0.89474 | 0.88154 |
| LR | 0.73638 | 0.99823 | 0.47368 | 0.64246 |

During the training process of WGAN-GP, we incorporated the CTST process after each epoch to promptly assess the impact of network training and retain the necessary network weights for downstream tasks. Due to the time-consuming nature of network training, cross-validation is performed only once, and the network weights are retained based on the accuracy of this validation. However, these weights are susceptible to chance during the individual CTST process. This is because, due to the probabilistic nature of data generation, the network may generate samples that closely resemble a specific distribution, leading to a CTST result close to 0.5. (In reality, the network has already learned the distribution of the target samples more effectively at this stage.) These weights can significantly influence the downstream task. To evaluate the impact of generator fitting across different epochs, we conducted an additional CTST experiment. Multiple CTSTs were performed using different weights, and the classifier with the mean accuracy closest to 0.5 was selected as the data generator for the downstream task.

A total of 61 weights were saved using the preset hyperparameters. The additional CTST experimental results for these 61 weights are presented in Figure 2, sorted based on their variance. After conducting 200 repetitions of the experiment, slight differences were observed in the mean values of all generator test results. Most of the generator test results still satisfy the preset hyperparameter criteria, falling between 0.498 and 0.502 as denoted by the green dots in the figure. A few generator test results deviate from the preset criteria but meet the requirements of the downstream tasks, ranging from 0.4 to 0.498 and 0.502 to 0.6 as indicated by the orange points in the figure. This indicates an increase in their variance. A few generators, represented by the red dots, exhibit test results significantly deviating from the 0.5 criterion, with an absolute difference greater than 0.1, and higher variance. In such cases, the samples generated by these generators can introduce bias in the training of the classifier model. These test results confirm necessity of adding additional CTST.

图表, 散点图

描述已自动生成

Figure 2: CTST result over 61 saved model weight.

**3.3.WGAN-GP successfully generated high-quality positive gene samples**

Overall, the samples synthesized through WGAN-GP exhibited a strong alignment with the original positive samples. Following each training epoch, we incorporated supplementary CTST. As shown in figure 3, at the outset of generator training, the generated samples diverged notably from the distribution of positive samples, yielding a CTST accuracy of 1.0. This observation underscores the effective discrimination achieved by the CTST classifier between the generated and positive samples.

Around the 200th epoch, the generator began capturing the distinctive feature distribution of the positive samples. Consequently, an overlap emerged in the feature space between the generated and positive samples, leading to a decrease in CTST accuracy to approximately 0.7. As training progressed, by the 500th epoch, this overlapping region expanded further, causing the CTST accuracy to diminish to about 0.6.

After over 800 additional training iterations, WGAN-GP successfully internalized the genetic feature distribution of authentic positive samples. Evidently, the CTST accuracy decreased to 0.5 and exhibited fluctuation around this value, signifying the generator's ability to effectively align with the positive sample distribution. Additionally, the training loss curve further indicated that the generator attained a local optimum as the CTST accuracy reached 0.5.

Furthermore, conducting independent CTST evaluations on each saved generator model unveiled that more than half of the generators could consistently uphold a classifier accuracy of around 0.5. This observation implies the challenge of distinguishing between generated samples and positive samples using the classifier. Undoubtedly, the generator excelled at producing high-caliber synthetic samples.

图表, 散点图

描述已自动生成

Figure 3: Distribution changes between positive and synthetic samples

During the early stages of developing generative adversarial networks, researchers were amazed by GAN's remarkable fitting ability, but also faced the challenge of training difficulties. While WGAN-GP partially addressed these difficulties by incorporating gradient penalties, the role of proper neural network design and hyperparameter tuning remained crucial in GAN training. In addition to visualizing samples at various epochs, observing changes in loss during GAN training was also effective. For WGAN-GP, the generator's loss represented the disparity between the generated sample and the real sample. Figure 4 illustrates that the generator exhibited poor sample fitting ability at the beginning of training, resulting in an increase in loss throughout the training process. After adversarial training with the discriminator, the generator started capturing the underlying sample distribution, leading to a decrease in loss. As the training progressed, the generator's loss began to fluctuate, indicating its position in the intermediate stage. Around 500-600 epochs, the generator and discriminator reached a Nash equilibrium in the game, demonstrating a consistent reduction in loss and subsequent stabilization. This indicated that the entire generative adversarial network had reached an approximate optimal state.

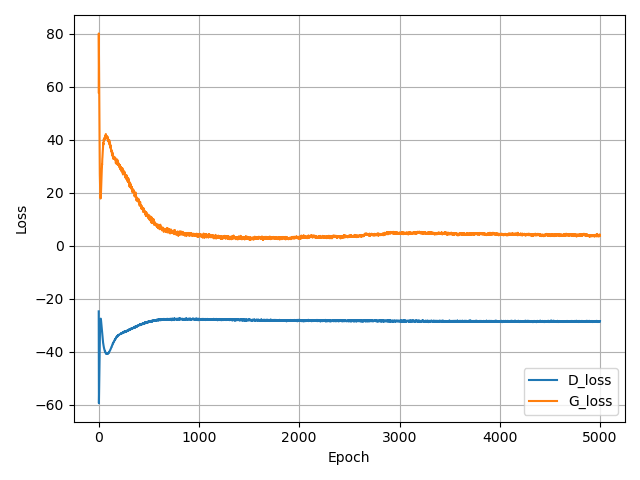


Figure 4: WGAN-GP train loss over 5000 epochs. Blue line indicates discriminator loss while orange line indicates generator loss.

**3.4. Performance comparison**

To demonstrate the predictive performance of adding WGAN-GP synthesized samples to the original samples, we compared it with SMOTE and used no oversampling as the baseline. The oversampling scheme involved oversampling five different machine learning classifiers during training, instead of balancing the entire dataset before training. In order to comprehensively evaluate the classifier's performance, we used five evaluation indicators: MCC, accuracy, precision, recall, and F1 score. Due to the significant imbalance in the proportion of positive and negative samples in the test set, we conducted block processing on the samples during testing. This process was similar to the training process but did not involve oversampling. Each classifier received multiple models after training, and we used all the trained models to test each test data block. The average of all test results was used as the classifier's performance.

As illustrated in the table 3, our scheme outperforms other schemes in most indicators. First, when considering classifiers, our proposed oversampling scheme effectively enhances the performance of all five classifiers. In the case of SVM classifiers, using WGAN-GP for oversampling led to an improvement in accuracy by approximately 2% and 1% compared to the unsampled scheme and the SMOTE scheme, respectively. It also increased precision by about 4.6% and 1%, and the F1 score by approximately 2.6% and 1.7%. Regarding MCC, a common measure for sample imbalance in classification tasks, our experiments showed an increase of about 5.9% and 2%, respectively. The performance of the other four classifiers under various oversampling schemes is similar to that of SVM, but the performance improvement for some classifiers under different oversampling schemes is less significant. Second, when considering performance evaluation indicators, although our proposed scheme doesn't exhibit significant improvements in some indicators for certain classifiers, across all test results, our method achieves the optimal values for all evaluation indicators (which may not originate from the same classifier). In terms of accuracy, the RF classifier achieves the highest accuracy of 93.199%, precision of 93.846%, recall of 97.502%, F1 score of 93.274%, and MCC of 86.592% in our scheme.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Classifier | Oversample Method | Accuracy | Precision | Recall | F1 | MCC |
| SVM | WGAN-GP | 92.189 | 93.340 | 91.228 | 92.148 | 84.619 |
| SMOTE | 91.080 | 92.341 | 90.039 | 90.854 | 82.608 |
| None | 89.164 | 87.761 | 91.780 | 89.539 | 78.766 |
| RF | WGAN-GP | **93.199** | 93.476 | 93.280 | **93.274** | **86.592** |
| SMOTE | 92.818 | 93.298 | 92.685 | 92.854 | 85.864 |
| None | 87.967 | 86.473 | 90.854 | 88.411 | 76.413 |
| GNB | WGAN-GP | 90.941 | 86.713 | **97.502** | 91.633 | 82.900 |
| SMOTE | 90.439 | 86.738 | 96.253 | 91.071 | 81.773 |
| None | 87.157 | 82.758 | 95.068 | 88.271 | 75.643 |
| GB | WGAN-GP | 92.325 | **93.846** | 90.960 | 92.201 | 84.952 |
| SMOTE | 92.179 | 93.829 | 90.663 | 92.074 | 84.645 |
| None | 89.663 | 88.518 | 91.893 | 89.981 | 79.760 |
| LR | WGAN-GP | 93.175 | 93.530 | 93.131 | 93.215 | 86.569 |
| SMOTE | 92.706 | 92.118 | 93.815 | 92.835 | 85.672 |
| None | 89.991 | 87.767 | 93.424 | 90.344 | 80.258 |

Table 3: Model performance comparison, measurement: %

**3.5**

**Discussion**

**References:**

**1. Liu, Yufei, et al. "Wasserstein GAN-based small-sample augmentation for new-generation artificial intelligence: A case study of cancer-staging data in biology." Engineering 5.1 (2019): 156-163.**

**2. Yin, Rui, et al. "ViPal: a framework for virulence prediction of influenza viruses with prior viral knowledge using genomic sequences." Journal of Biomedical Informatics 142 (2023): 104388.**

**3. Data augmentation and multimodal learning for predicting drug response in patient-derived xenografts from gene expressions and histology images**

**4. Predicting genes associated with RNA methylation pathways using machine learning**

5. Boccaletto, Pietro, et al. "MODOMICS: a database of RNA modification pathways. 2017 update." *Nucleic acids research* 46.D1 (2018): D303-D307.