Impacts of Data Preprocessing and Sampling Techniques on Solar Flare Prediction from Multivariate Time Series Data of Photospheric Magnetic Field Parameters

MohammadReza **EskandariNasab**
PhD Student at Utah State University

06/06/2024

UtahStateUniversity

Introduction

Project Type: **Research**

Reasons of this study:

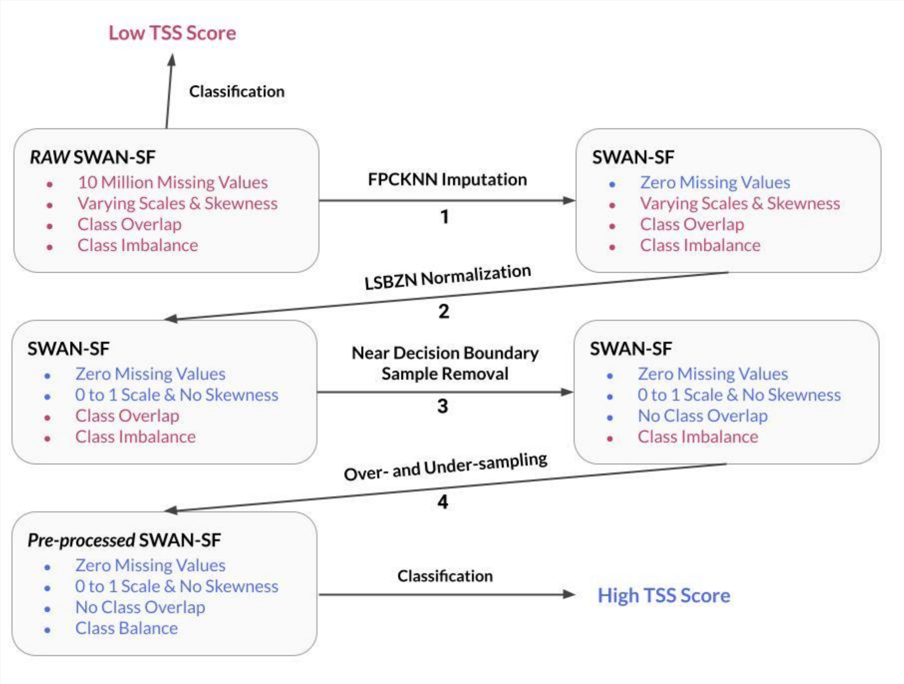
Exploring How Data Preprocessing and Sampling Techniques Enhance Classification Outcomes.

1. Implementing a Missing Value Imputation and Normalization technique for Multivariate Time Series Data.
2. Analyzing the Effects of Near Decision Boundary Sample Removal.
3. In-depth Analysis of Sampling Techniques Across 8 Classification Algorithms and 8 Methods.

Introduction

The study consists of four parts:

1. **FPCKNN Imputation**
2. **LSBZM Normalization**
3. **Near Decision Boundary Sample Removal**
4. **Over- and Under-sampling**



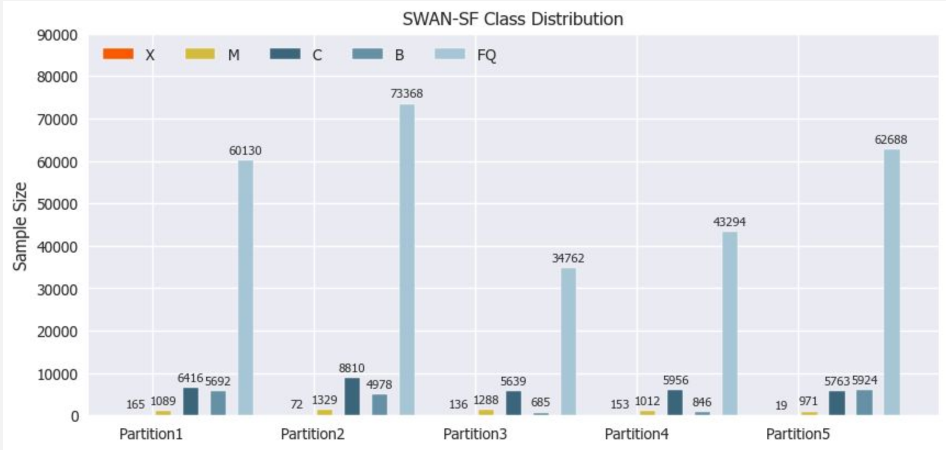
Dataset

SWANSF (5 flare classes: X, M, B, C, FQ)

- 5 different Partitions
- Solar flare data from 2010 to 2018
- 24 attributes, and 60 timestamps

- 4 different train-test combinations (In terms of temporal ordering, the training dataset should precede the testing dataset.)
- Two types of classification: **binary** and multiclass

https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/EBCFKM



Preprocessing

There are a few downsides to the SWAN-SF dataset:

- Over 10 million missing values
- Different scale of attributes and skewness
- Class overlap
- Class imbalance

They will result in low classification performance.

We introduced **FPCKNN Imputation** and **LSBZM Normalization** to tackle two of these problems.

Table 2. Missing Value Distribution in SWAN-SF Dataset

Attribute	Partition 1	Partition 2	Partition 3	Partition 4	Partition 5
Total Null	2487146	4002503	1472395	1900777	2990768
Total Not-Null	107750854	128832997	62292605	74990723	110056732
R_VALUE	2399220	2934918	1361095	1748394	2755911
TOTUSJH	652	93300	2718	4844	4964
TOTBSQ	652	93300	2718	4844	4964
TOTPOT	652	93300	2718	4844	4964
TOTUSJZ	652	93300	2718	4844	4964
ABSNJZH	652	93300	2718	4844	4964
SAVNCPP	652	93300	2725	4844	4964
USFLUX	652	93300	2718	4844	4964
TOTFZ	652	93300	2718	4844	4964
MEANPOT	0	0	0	0	0
EPSX	0	0	0	0	0
EPSY	0	0	0	0	0
EPSZ	0	0	0	0	0
MEANSHR	0	0	0	0	0
SHRGT45	81406	134585	84113	103943	185217
MEANGAM	0	0	0	0	0
MEANGBT	0	0	0	0	0
MEANGBZ	0	0	0	0	0
MEANGBH	0	0	0	0	0
MEANJZH	0	0	0	0	0
TOTFY	652	93300	2718	4844	4964
MEANJZD	0	0	0	0	0
MEANALP	0	0	0	0	0
TOTFX	652	93300	2718	4844	4964

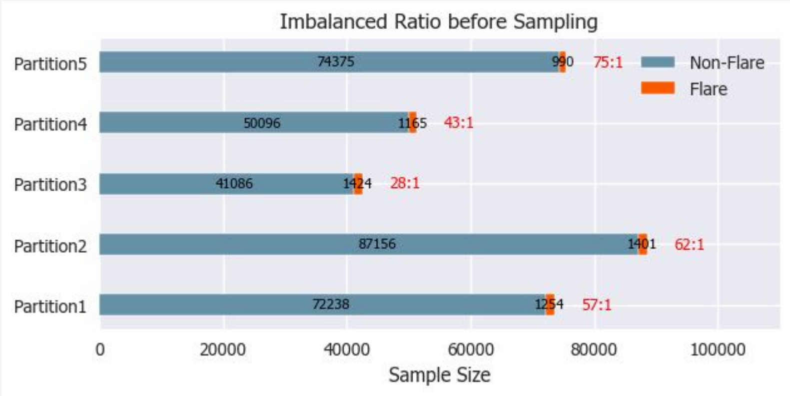
Sampling

The SWAN-SF dataset is imbalanced, indicated by a significantly lower number of samples in the 'Flare' class compared to the 'Non-Flare' class.

To tackle this problem we need to take advantage of sampling techniques.

But which sampling technique is good for SWAN-SF?

Will doing sampling improve the performance of classification?



Sampling

We studied the impact of sampling techniques on SWAN-SF:

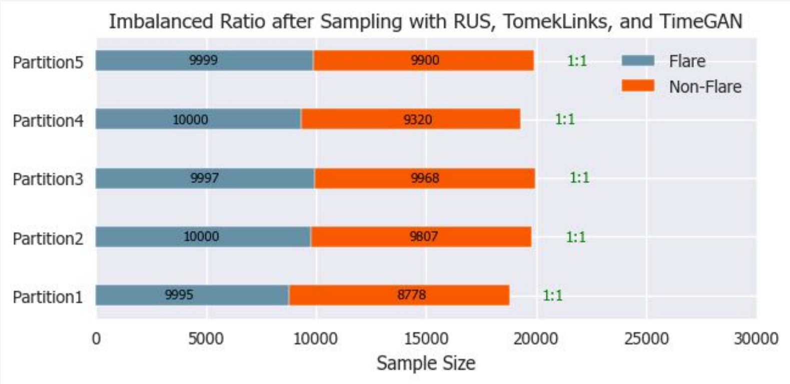
Over-sampling:

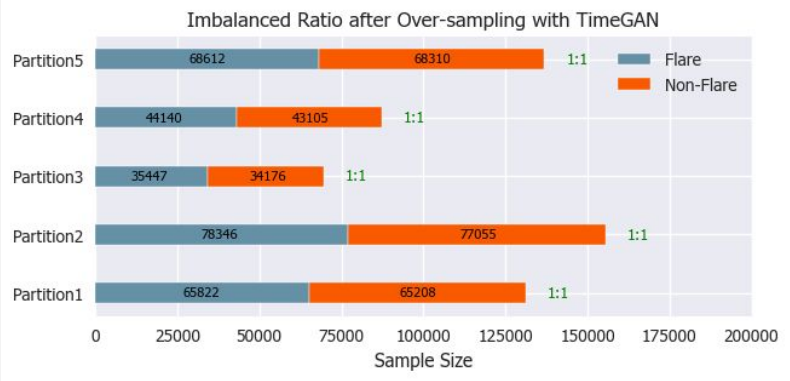
SMOTE, ADSYN, Gaussian Noise Injection, TimeGAN.

Results in high number of samples. Therefore:

Combination of Over and Under-sampling:

1. RUS, Tomek-Links, SMOTE
2. RUS, Tomek-Links, ADASYN
3. RUS, Tomek-Links, GNI
4. RUS, Tomek-Links, TimeGAN





Sampling

Smote (Synthetic Minority Oversampling Technique): Available in ***imblearn*** library

Specifically, a random example from the minority class is first chosen. Then k of the nearest neighbors for that example are found (typically k=5). A randomly selected neighbor is chosen and a synthetic example is created at a randomly selected point between the two examples in feature space.

Adasyn (Adaptive Synthetic Sampling): Available in ***imblearn*** library

It calculates the density distribution of each minority class sample and generates synthetic samples according to the density distribution. This adaptive approach ensures that more synthetic samples are generated for minority class samples that are harder to learn, thus improving the classification performance of machine learning models.

Sampling

Gaussian Noise injection: Implementable by Python

Gaussian noise injection works by adding random values from a Gaussian (normal) distribution to your data. Here's a more detailed breakdown of how it works.

I applied a noise range equal to 5% of the standard deviation for the added noise.

```
std_dev = np.std(X_train, axis=0)

noise_level = std_dev * noise_proportion

noise = np.random.normal(0, noise_level, sample.shape)

 new_sample = sample + noise
```

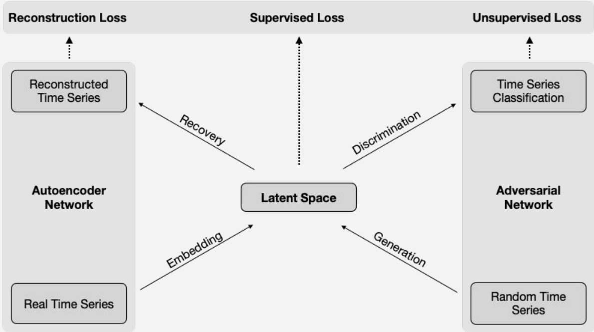
Sampling

TimeGAN (Time Series generative Adversarial network): Available on my GitHub (Compatible with TF2)

It's an adaptation of the traditional Generative Adversarial Network (GAN) framework, tailored to handle the unique characteristics of time series data.

TimeGAN is compatible with Multivariate Time Series data, so that

we can feed a 3D dataset to the function.

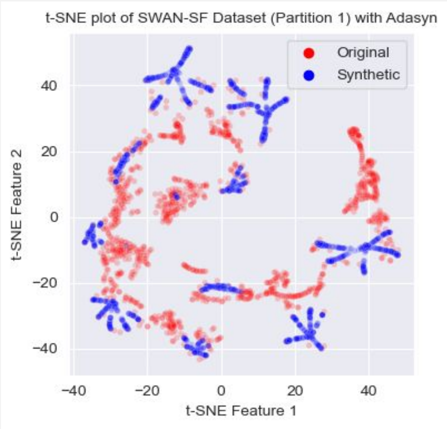


from timegan import timegan

generated_data = timegan(minority_class_data, parameters, *num_of_data_to_be_generated*)

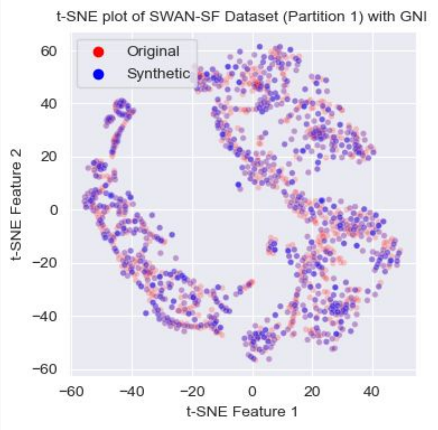
https://github.com/samresume/TimeGAN-TF2_Compatible

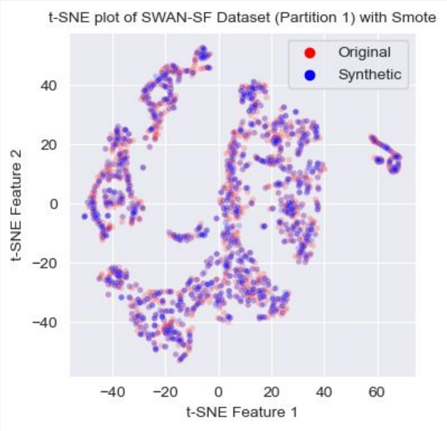
Sampling



Adasyn

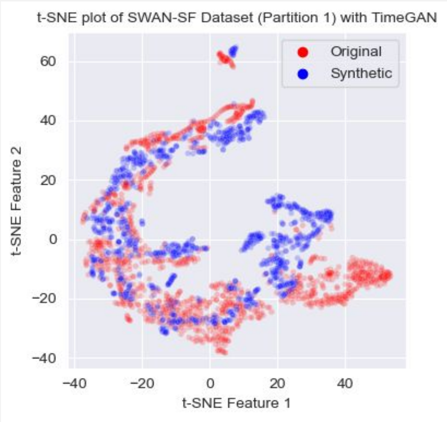
GNI





Smote

TimeGAN



Experiments

We have Two approaches:

- Classification on new Statistical Features:

Statistical Features: **First_Value, Last_Value, Mean, Median, Weighted_Avg, STD, Skewness, Kurtosis, Slope**

Methodes: **SVM, MLP, KNN, Random Forest**

- Classification on actual Time Series:

Methodes: **LSTM, CNN, GRU, RNN**

Experiments

Training Dataset :Imputation, Normalization, Removing C Class (Overlap), Sampling

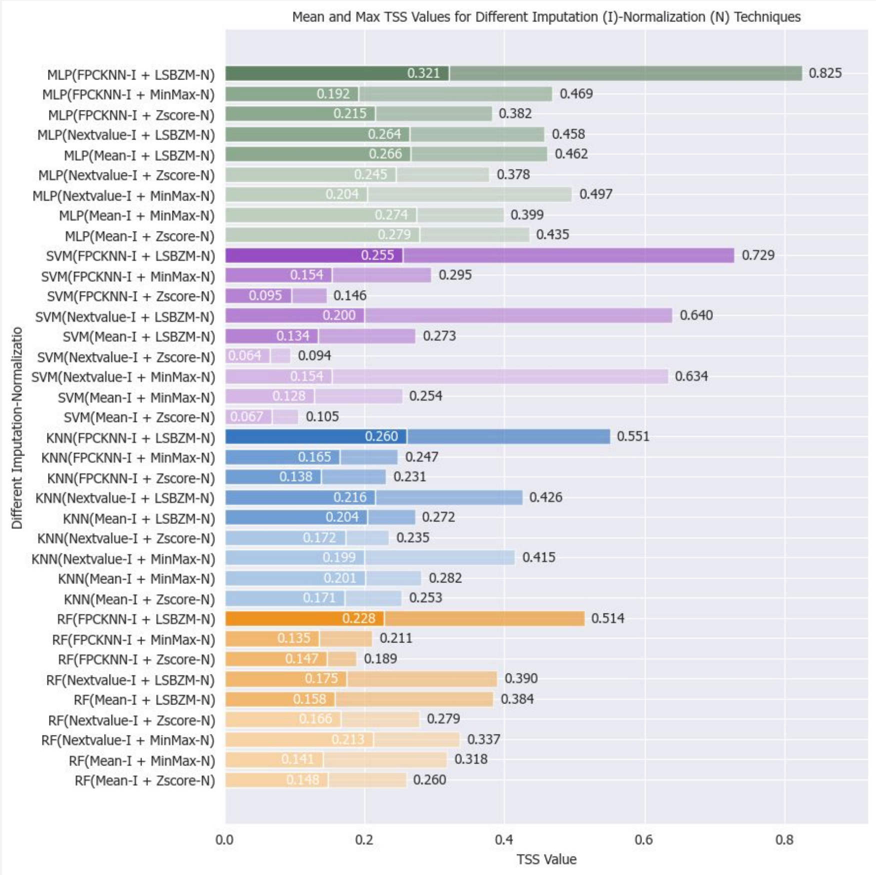
Test Dataset: Imputation, Normalization

We have 10 different train-test splits:

(1,2), (2, 3), (3,4), **(4,5)**

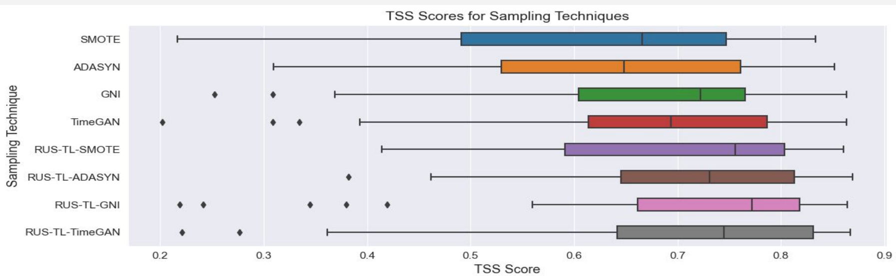
Experiments

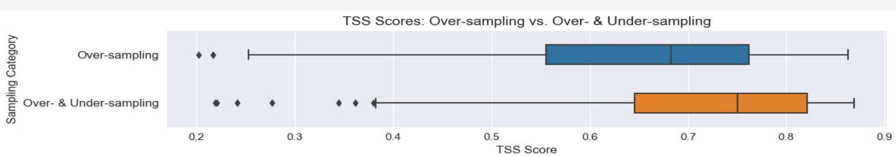
The results of **Imputation and Normalization**

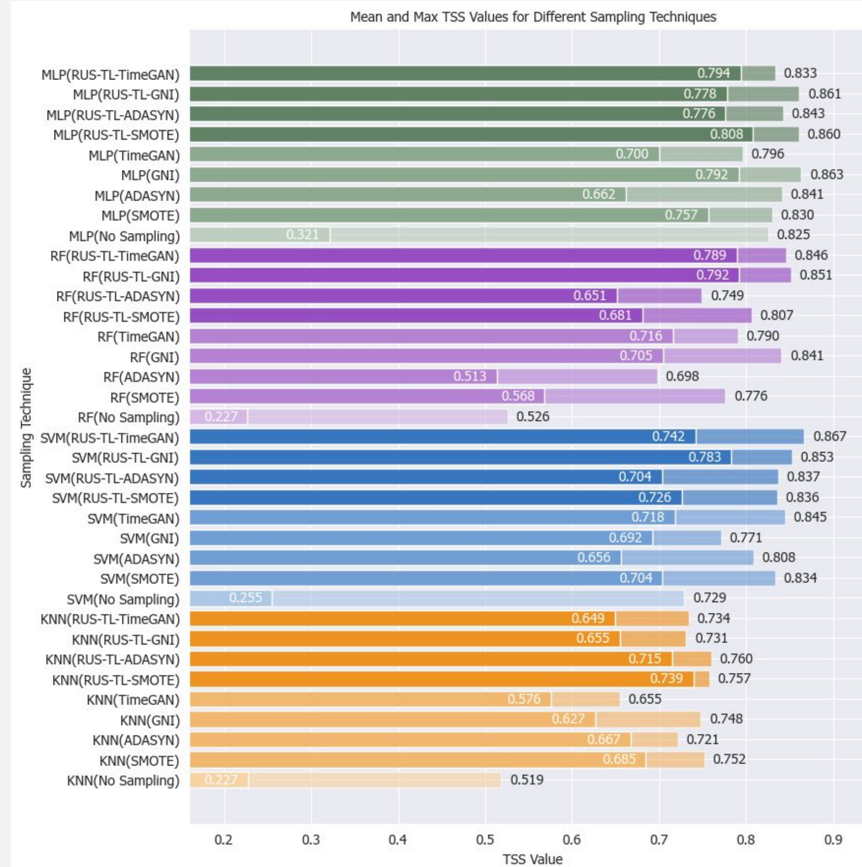


Experiments

The results of **Sampling**







Experiments

- Combination of Over and Under-sampling techniques are the best.
- TimeGAN and SMOTE are the best Over-sampling techniques for SWAN-SF
- Adasyn is the worst Over-sampling technique for SWAN-SF
- A proper Normalization technique is a crucial step for getting high classification performance
- MinMax normalization is better than Z-Score normalization for SWAN-SF

Mean TSS before any preprocessing : **0.1 to 0.3**

Mean TSS after our Imputation and Normalization Technique: **0.5 to 0.75**

Mean TSS after all Four Parts: **0.75 to 0.9**

Conclusion

- Enhancing the performance of a classification task significantly depends on precise preprocessing and sampling methods.

- The choice of effective imputation and normalization techniques, along with appropriate sampling strategies, can notably influence the overall performance

https://github.com/samresume/Imbalanced-Solar-Flare-Prediction-on-SWANSF

Top References

Yoon, J., Jarrett, D., & Van der Schaar, M. (2019). Time-series generative adversarial networks. Advances in Neural Information Processing Systems, 32.

Bobra, M. G., & Couvidat, S. (2015). Solar Flare Prediction Using SDO/HMI Vector Magnetic Field Data with a Machine-Learning Algorithm. The Astrophysical Journal, 798(2), 135. https://dx.doi.org/10.1088/0004-637X/798/2/135

Ahmadzadeh, A., Aydin, B., Georgoulis, M. K., Kempton, D. J., Mahajan, S. S., & Angryk, R. A. (2021). How to Train Your Flare Prediction Model: Revisiting Robust Sampling of Rare Events. The Astrophysical Journal Supplement Series, 254(2), 23. https://dx.doi.org/10.3847/1538-4365/abec88

Angryk, R., Martens, P., Aydin, B., Kempton, D., Mahajan, S., Basodi, S., Ahmadzadeh, A., Cai, X., Filali Boubrahimi, S., Hamdi, S. M., Schuh, M., & Georgoulis, M. (2020). Multivariate Time Series Dataset for Space Weather Data Analytics. Scientific Data, 7, 227. https://doi.org/10.1038/s41597-020-0548-x

Thank You for Your Attention

MohammadReza **EskandariNasab**
PhD Student at Utah State University

06/06/2024

UtahStateUniversity