Impact of Noise on Machine Learning Algorithms Performance

How noise in datasets affect the performance of ML algorithms

Neha Tiwari
MSc Computer Science - Data Science
Trinity College Dublin
Dublin, Ireland
tiwarin@tcd.ie

Krishna Hariramani
MSc Computer Science - Data Science
Trinity College Dublin
Dublin, Ireland
hariramk@tcd.ie

Ankur Rangwala
MSc Computer Science - Data Science
Trinity College Dublin
Dublin, Ireland
rangwala@tcd.ie

INTRODUCTION

Noise (in the data science space) is unwanted data items, features or records which don't help in explaining the feature itself, or the relationship between feature and target [6]. Noise often causes the algorithms to miss out patterns in the data. In order to train a model, we need to filter the noise, since noise can affect algorithms performance adversely. Hence, knowing the exact impact of noise in our dataset is an important issue and should be evaluated to develop an effective model to make a better decision accordingly.

RELATED WORK

Noise in dataset affects the performance of ML algorithms and thus motivated the study related to different ways of creating and introducing noise in the dataset to measure its impact on different ML algorithms [2]. Research done on noise can be categorized in two fields.

1. Noise generation:
There are different ways to characterize noise generation. Noise can be characterized by either its distribution (Normal (Gaussian)) or where the noise is introduced (individual or combination of training and test data or input attributes or output class) or differentiating whether the generated values are related to each data value or min, max and standard deviation of each feature.

2. Impact of noise in algorithms:
In [3], Zhu investigated different types of noise in training and test data, input attributes, output class and identified a comprehensive analysis of noise using the C4.5 algorithm.

METHODOLOGY

Figure 1 and 2 shows the workflow diagrams for datasets 1 and 2 respectively at a macro level.

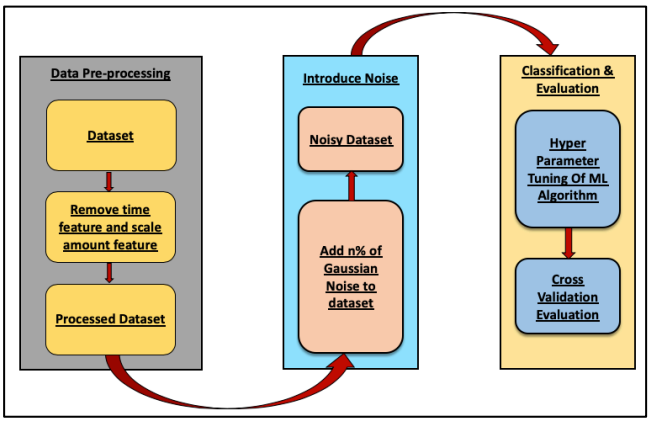

Figure 1: Dataset 1 workflow diagram.

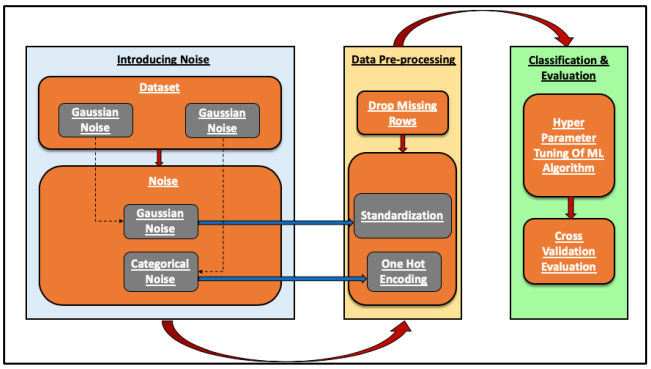

Figure 2: Dataset 2 workflow diagram

1. Dataset and Pre-Processing

Dataset used:

- Credit card fraud.
- Census income data set.

Dataset 1 (credit card fraud) - We have used credit card fraud dataset from Kaggle [4] with 284807 records of transactions (made in September 2013 over two days) and 28 normalized numerical features, two additional features 'Time' and 'Amount' are also

provided. We dropped the Time feature as it does not give any information for labels, we normalized and scaled the 'Amount' feature as it can give information like the amount of fraud. Finally, we get 29 scaled and normalized numerical features and one label column 'Class' containing 0's and 1's (1 denoting fraud transaction).

Dataset 2 (census income dataset) – We have used census income dataset from UCI machine learning repository [7]. After dropping the rows with missing values, the dataset has 30162 rows and 15 columns out of which 22654 were negative class. In this dataset, we have used ***'one hot encoding'*** for categorical features and ***'standardized'*** the numerical features. We updated our label 'salary' by replacing values <=50k $ with 0's and >50k $ with 1's.

2. Introduce Noise to the dataset

We have used different techniques to introduce noise for both dataset.

For credit-card fraud data set we have added noise to dataset using gaussian noise. For adding n% of noise to the dataset, we use the algorithm shown in Figure 2[5]. Figure 3 shows an example for the same.

Find the standard deviation SD of each feature f
Pick n% of random rows from the original dataset and store in a new variable ND
FOR each instance in ND,
    FOR each feature value of feature f,
        1. Choose a random number x taken from the interval (-SD, SD)
        2. Add x to that instance
CONCATINATE ND at the end of original dataset
SHUFFLE all dataset and Randomly SELECT number of rows in the original dataset

Figure 2: Algorithm for adding noise in dataset1

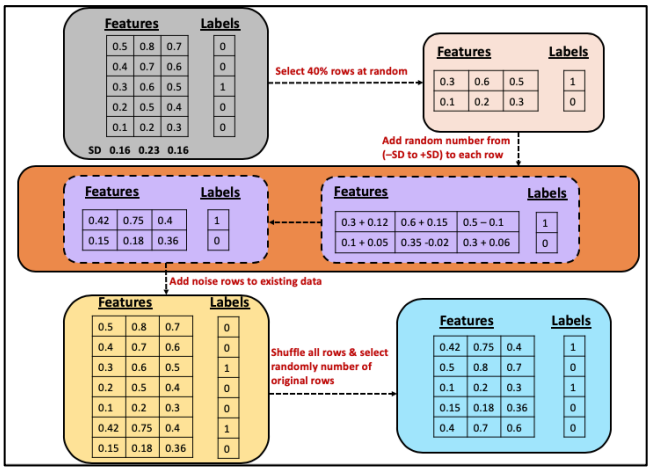

Figure 3: Example of adding 40% noise to dataset1

For census income dataset we have added noise to numerical features using gaussian noise[5]. We introduced noise in categorical features by replacing original category values with randomly selecting a category value from all the categories present in that feature. While selecting random values, we do not consider the original value. For adding n% of noise to the dataset, we use the algorithm shown in Figure 4. Figure 5 shows an example for the same.

Find the standard deviation SD of each feature f
Pick n% of random rows from the original dataset and store in a new variable ND
FOR each instance in ND,
    FOR each numerical feature f,
        1. Choose a random number x taken from the interval (-SD, SD)
        2. Add x to that instance
    FOR each categorical feature f,
        1. Choose randomly any value of category in that categorical feature other than original
        2. Replace original category value to randomly selected value
CONCATINATE ND at the end of original dataset
SHUFFLE all dataset and Randomly SELECT number of rows in the original dataset

Figure 4: Algorithm for adding noise in dataset2

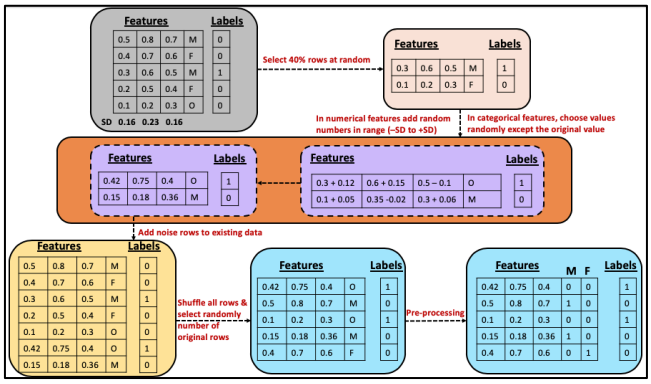

Figure 5: example of adding 40% noise in dataset2

3. Classification and Evaluation

For classification, we use Logistic Regression (LR), Gaussian Naive Bayes Classifier (GNBC) and Random Forest Classifier (RFC) with tuning hyper-parameters. Table 1, shows the hyper parameters chosen for each algorithm after running 10 folds cross-validation. We have used ***'scikit-learn'*** library in python [8] for machine learning algorithms, cross validation and hyper-parameter tuning.

ML algorithm	Hyper-parameters considered	Optimal Hyper-parameters for dataset 1	Optimal Hyper-parameters for dataset 2
LR	C: 0.01, 0.1,1,10,100 Penalty: 'l1', 'l2'	C: 0.01 Penalty: l1	C: 100 Penalty: l1
GNBC	No hyper-parameter required	-	-
RF	Bootstrap: true, false. max_depth: 50, 75, 100 n_estimators: 10,30,50	Bootstrap: true. max_depth: 50. n_estimator: 30	Bootstrap: true. max_depth: 100. n_estimator: 50

Table 1: Hyper parameters

Evaluation metrics for dataset 1:

Since dataset 1 is highly unbalanced, the negative class (frauds) account for only 0.172% of all transactions, evaluation metrics like accuracy and AUC under ROC should not be considered for evaluating the models. As we are interested in correctly predicting the frauds, we have used negative predicted value (NPV) /recall for negative class (TN/TN+FN) as a metric to evaluate our model's performance. We are also showing specificity (TN/TN+FP) at different noise levels. We have used 10 fold cross validation to calculate both specificity and NPV.

Evaluation metrics for dataset 2:
Dataset 2 is unbalanced, the negative class (salary <50k $) account for only 75% of all records, so we have used 10 fold cross validation to calculate accuracy, precision, recall and f1 score, at different noise levels.

RESULTS & DISCUSSIONS

Results of our experiments on different algorithms are discussed below.

1. Logistic Regression

Figure 6 depicts that value of NPV for negative class follows a decreasing trend (57% to 14%) with an increase in noise level (0% to 90%). Also, we can see that specificity does not follow any trend in general due to class imbalance.

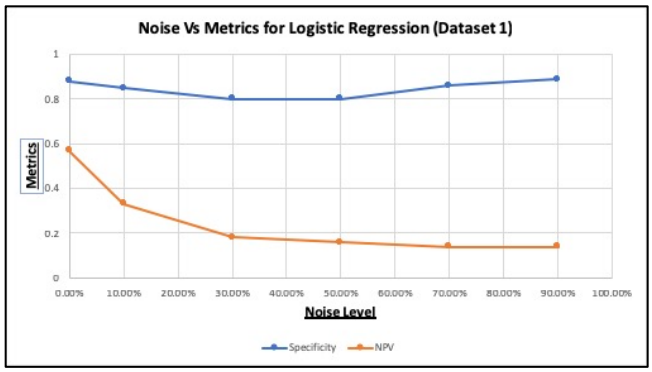

Figure 6: Noise Vs Metrics for Logistic Regression (Dataset 1)

Figure 7 depicts that with an increase in noise level (0% to 90%). All the metrics (accuracy, precision, recall and f1 score) follow decreasing trend, but the fall in all the metrics is very small.

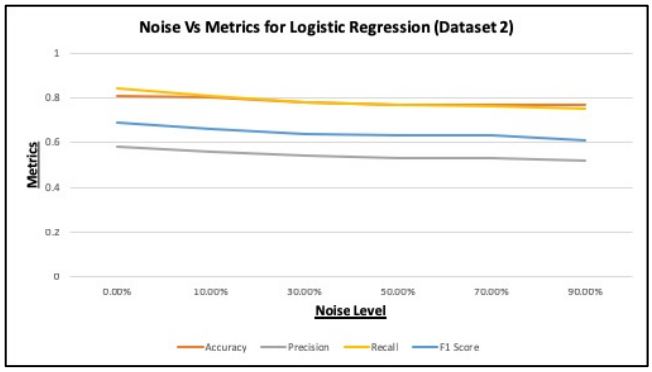

Figure 7: Noise Vs Metrics for Logistic Regression (Dataset 2)

2. Gaussian Naive Bayes (GNB) classifier

Figure 8 shows the performance metrics at different noise levels. We can see, very low values of specificity but values of NPV are better than logistic regression. Also as noise increases (0% to 90%), decrease in NPV (83% to 37%) is more robust than logistic regression, but this robustness to noise in NPV comes at the cost of low overall specificity.

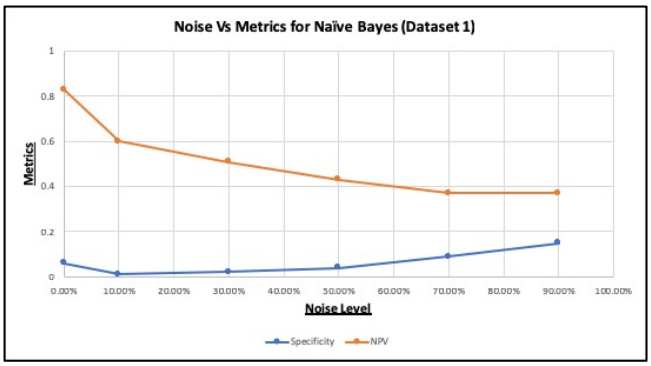

Figure 8: Noise Vs Metrics for GNB (Dataset 1)

Figure 9 shows the performance metrics at different noise levels. We can see, very low values of f1 score and recall, but values of accuracy and precision are better than logistic regression. Also as noise increases (0% to 90%), decrease in accuracy and precision is more robust than logistic regression, but this robustness to noise in accuracy and precision comes at the cost of low overall f1 score and recall.

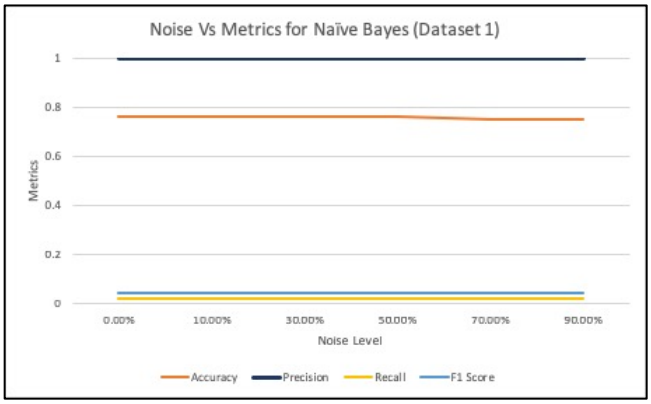

Figure 9: Noise Vs Metrics for GNB (Dataset 2)

3. Random forest (RF) classifier

Figure 10 illustrates that as noise increases, NPV decreases from 77% to 41% whereas specificity at all noise levels is good and does not follow any trend.

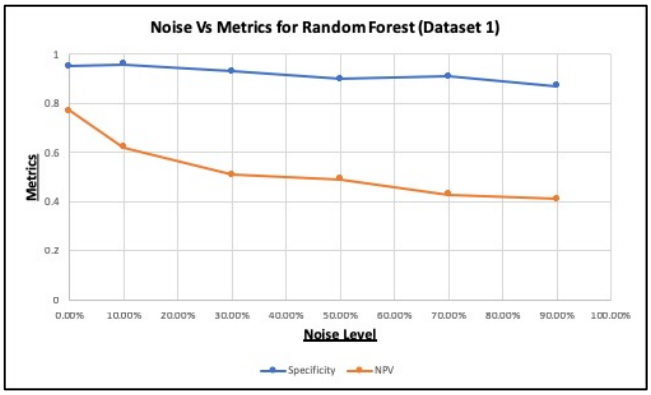

Figure 10: Noise Vs Metrics for RF (Dataset 1)

Figure 11 illustrates that as noise increases, recall and f1 follows a decreasing trend whereas precision and accuracy at all noise levels are good and do not follow any trend. Also, robustness of decrease in recall and f1 to noise is better than logistic regression.

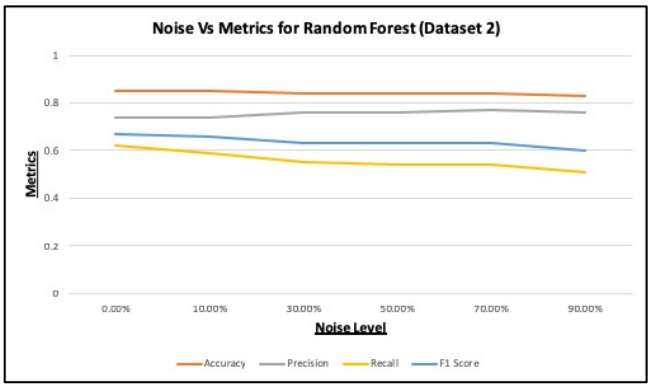

Figure 11: Noise Vs Metrics for RF (Dataset 2)

From above experiments, we conclude that:
1. Logistic Regression is the least robust to noise.
2. GNB classifier is most robust to noise but provides low specificity in dataset 1 and low recall and f1 score in dataset 2.
3. RF classifier is more robust to noise than Logistic Regression but less robust than GNB classifier. Also, it maintains high specificity throughout different noise levels in dataset 1 and in dataset 2 it maintains high recall and f1 score.

Figure 12 shows specificity and NPV for all the three algorithms on dataset 1.
Figure 13 shows accuracy and f1 score for all the three algorithms on dataset 2.

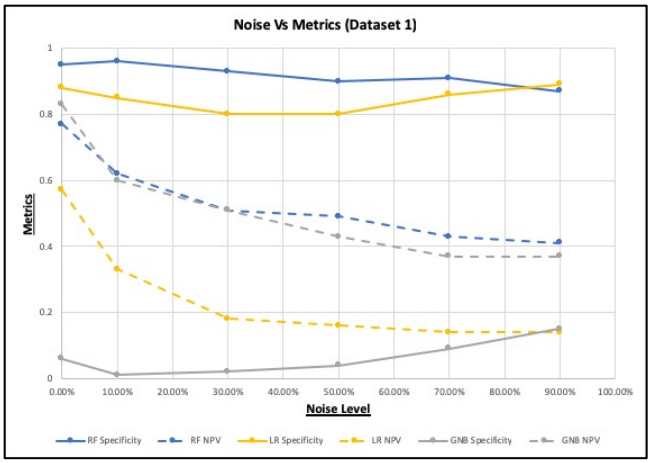

Figure 12: Noise vs Metrics for dataset 1

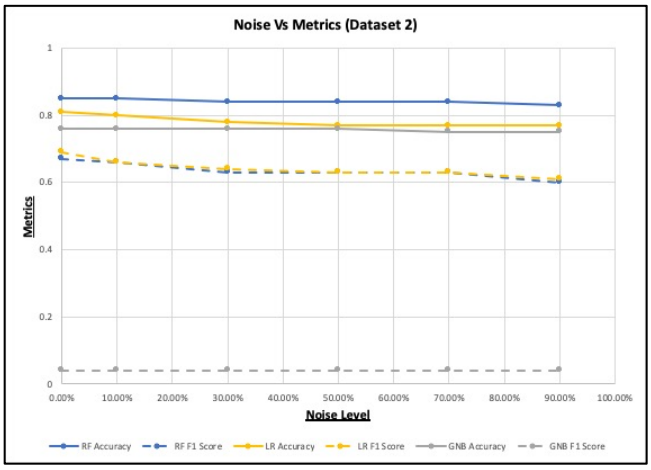

Figure 13: Noise vs Metrics for dataset 2

LIMITATION & OUTLOOKS

Due to time constraints, currently, we have evaluated the impact of noise on only three ML algorithms (GNB classifier, Logistic Regression and RF classifier). Also, we did not tune all the parameters for Random forest algorithm and did not tune hyper-parameters for each noise level. However, we would try to investigate the impact after performing hyper-parameter tuning for each noise level, in addition to experimenting with other algorithms like decision tree and k-nearest neighbors (KNN). We would also like to investigate effect of other forms of noise on the machine learning algorithms as part of the future work.

REFERENCES

[1] Kalapanidas, Elias, Nikolaos Avouris, Marian Craciun, and Daniel Neagu. "Machine learning algorithms: a study on noise sensitivity." In *Proc. 1st Balcan Conference in Informatics*, pp. 356-365. 2003.

[2] Nettleton, David F., Albert Orriols-Puig, and Albert Fornells. "A study of the effect of different types of noise on the precision of supervised learning techniques." *Artificial intelligence review* 33, no. 4 (2010): 275-306.

[3] Zhu, Xingquan, and Xindong Wu. "Class noise vs. attribute noise: A quantitative study." *Artificial intelligence review* 22, no. 3 (2004): 177-210.

[4] Bozsolik, Timo, Credit Card Fraud Detection, Retrieved from https://www.kaggle.com/mlg-ulb/creditcardfraud

[5] Lee, T., 2015. How to add some noise data to my classification datasets. Retrieved from https://www.researchgate.net/post/How_to_add_some_noise_data_to_my_classification_datasets.
[6] Ankit Rathi, Dealing with Noisy Data in Data Science, Retrieved from https://medium.com/analytics-vidhya/dealing-with-noisy-data-in-data-science-e177a4e32621
[7] Ronny Kohavi and Barry Becker, Census Income Data Set, retrieved from https://archive.ics.uci.edu/ml/datasets/census+income
[8] Open Source, scikit-learn, https://scikit-learn.org/stable/