# Quick look into Stochastic Variational inference using Variable Auto-Encoding for Credit Card dataset


## Abstract

In day-to-day operations, the dataset we get from banks, the medical industry, the retail house market, the stock market, and other environments don't have a proper balance between labels/classes. So, for example, in Credit card fraudulent systems, more than 99.9% of transactions are non-fraudulent; only 0.1% or less of transactions might be fraudulent or invalid. Therefore, training an ML model from this dataset will not perform as desired, as the model becomes more biased towards the valid transactions and will predict many fraudulent transactions as accurately. To overcome this, we need to provide the model with balance class labels, but acquiring these labels is very difficult. In this research, we will implement a flavor of auto-encoders called Variable Auto Encoders (VAE), which can generate more data points to make the distribution of the dataset even.


## Introduction

Traditional autoencoders are trained by taking input data and outputting the same data back out again. In the process, the number of dimensions increases and decreases. Variational autoencoders add additional randomness to it, so the output data isn't necessarily the same as the input data. This allows for better coverage of sparse data sets. Another advantage is that the decode phase, which takes normally input from the encoder and some randomness, can be used separately to generate similar data to the dataset. This works because the representation of the data output from the encoder is dense. The decoder takes this dense data and returns it to sparse data, similar to the original dataset.

XGBoost is an ensemble model consisting of many different decision trees. Each model is trained on a weighted sample, where the most inaccurate samples have a higher weight. It utilizes the Gradient Boosting Framework and also is fully parallelized.

Logistic regression works by assuming the probability of a specific outcome is the linear combination of the log of each likelihood. The function that converts log odds to a possibility is the logistic function. As the output of the logistic function increases rapidly in the center with the derivative tapering off at each extremum, the logistic function works well for binary classification.

## Literature Survey

### Stochastic Variational Inference

A stochastic variational inference is a scalable algorithm for approximating posterior distributions that can easily handle large data sets, outperform traditional variational inference methods, and develop a large class of probabilistic models. Stochastic inference can be applied to complex Bayesian models to

massive data sets. The algorithm constructs on variational inference, which transforms complex inference problems into high-dimensional optimization problems. It is optimization to iterate between re-analyzing each data point in the dataset and re-estimate its hidden structure. In the experiment section, it is applied to two probabilistic models, latent Dirichlet allocation and hierarchical Dirichlet process topic model, thus showing it can efficiently analyze massive data sets with complex probabilistic models

## Fast Inference in Sparse Coding Algorithms with Applications to Object Recognition

An algorithm called Predictive Sparse Decomposition (PSD) is proposed simultaneously to learn an overcomplete linear basis set; it produces a predicator easy compute approximator that predicts the optimal sparse representation. The experiments show that the predictor is over 100 times faster than the available fastest sparse optimization algorithm. Also, it produces features that produce better recognition accuracy on visual object recognition tasks compared to optimal optimization representations. A feedforward regressor can approximate sparse codes without compromising the recognition accuracy, making the recognition process very fast and suitable for real-time systems. It can be used as a pre-processor in many vision applications and to extract features in object recognition systems. To our knowledge, no sparse coding algorithm is computationally efficient because inference involves some iterative optimization.

## Black Box Variational Inference

The “Black box” variational inference algorithm can be quickly applied to many models with slight additional derivation. Based on a stochastic optimization of the variational objective, the noisy gradient is computed from Monte Carlo samples from the variational distribution. Develop several methods to reduce the variance of the gradient, always maintaining the criterion that we want to avoid complicated model-based derivations. The experiment evaluates our approach against the corresponding black box sampling-based methods. Our method reaches better predictive likelihoods much faster than sampling methods. Black Box Variational Inference lets us easily explore a vast space of models by quickly constructing and evaluating several models of longitudinal healthcare data.
The primary approach is a stochastic optimization of the ELBO by sampling from the variational posterior to compute a noisy gradient. Essential to its success are model-free variance reductions to reduce the variance of the noisy gradient. As a result, our method works well on new models while requiring minimal analytic work by the practitioner.

## An Analysis of Logistic Models: Exponential Family Connections and Online Performance

The two most analysis focus areas of logistic models are the relationship and exponential -based generative models and performance in an online and adversarial setting. The presented model resolves the long-standing relationship ambiguity by establishing a connection between a logistic model and an exponential generative model. Also, it is shown that online Bayesian logistic models are competitive with the best batch models, even in potentially adversarial settings for a given characterization of the family of conditional models that lead to affine log-odds of the posterior. For logistic regression models, we show that the log-odds ratio of the posteriors will be affine if and only if the conditional class distributions belong to the same exponential family. A similar result for structured prediction problems by showing that the label sequence posterior will be that of a dependent random field (CRF)if and only if the conditionals belong to structured exponential families. The second main result of this paper shows that an incrementally updated Bayesian logistic regression model will have performance comparable to that of the single best logistic regression classifier that can be obtained by training on accurate historical

data. The theoretical guarantee holds without any assumption about the data and strongly supports using Bayesian models for real-life online settings.

### The Helmholtz Machine

A way of finessing this combinatorial explosion is by maximizing an easily computed lower bound on the probability of the observations. The presented method can be viewed as a form of hierarchical self-supervised learning related to the function of bottom-up and top-down cortical processing pathways. The human perceptual system is viewed as a statistical inference engine whose role is to infer the probable causes of sensory input. We show that a device of this kind can learn how to perform these inferences without requiring a teacher to label each sensory input vector with its underlying causes. Instead, a recognition model is used to infer a probability distribution over the underlying causes from the sensory input, and a separate generative model, which is also learned, is used to train the recognition model. A connectionist system with multiple layers of neuron-like binary stochastic processing units connected hierarchically by two sets of weights. The Helmholtz machine is closely related to other schemes for self-supervised learning models that use feedback and feedforward weights. By contrast with adaptive resonance theory and the counter-streams model, the Helmholtz machine treats self-supervised learning as a statistical problem that is a generative model that accurately captures the structure in the input examples. The recognition process in the Helmholtz machine is a purely bottom-up-the top-down model with no direct role and no interaction between units in a single layer. However, such effects are essential for accurate perception. They can be implemented using iterative recognition, in which the generative and recognition activations interact to produce the final activity of a unit. This can introduce substantial theoretical complications in ensuring that the activation process is stable, converges adequately quickly, and in determining how the weights should change to capture input examples more accurately.

## Methodology

- Trained multiple models using VAE
- Run a sample LR model on the default dataset
- Check how good it is in predicting only faulty datasets.
- Pick records that failed to predict in the dataset.
- Generate new samples using VAE using 10 of the fault unpredictable datasets.
- Train a new LR model and repeat the steps.
- Note and evaluate the metrics.
- Test with multiple models of VAE.

## Dataset

We have a credit card dataset from https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud. By doing simple Exploratory data analysis, the following details are discovered.

## Columns

- Time
- V1 to V29 (PCA features with decimal values). These column names are obfuscated to protect critical information.
- Amount – Transaction carried out for each transaction
- Class – 0 Non-Fraudulent translation 1 Fraudulent transaction.

| Type | Time | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 |
|---|---|---|---|---|---|---|---|---|---|
| count | 284807 | 284807 | 284807 | 284807 | 284807 | 284807 | 284807 | 284807 | 284807 |
| mean | 94813.86 | 1.17E-15 | 3.42E-16 | -1.38E-15 | 2.07E-15 | 9.60E-16 | 1.49E-15 | -5.56E-16 | 1.21E-16 |
| std | 47488.15 | 1.958696 | 1.651309 | 1.516255 | 1.415869 | 1.380247 | 1.332271 | 1.237094 | 1.194353 |
| min | 0 | -56.4075 | -72.7157 | -48.3256 | -5.68317 | -113.743 | -26.1605 | -43.5572 | -73.2167 |
| 25% | 54201.5 | -0.92037 | -0.59855 | -0.89036 | -0.84864 | -0.6916 | -0.7683 | -0.55408 | -0.20863 |
| 50% | 84692 | 0.018109 | 0.065486 | 0.179846 | -0.01985 | -0.05434 | -0.27419 | 0.040103 | 0.022358 |
| 75% | 139320.5 | 1.315642 | 0.803724 | 1.027196 | 0.743341 | 0.611926 | 0.398565 | 0.570436 | 0.327346 |
| max | 172792 | 2.45493 | 22.05773 | 9.382558 | 16.87534 | 34.80167 | 73.30163 | 120.5895 | 20.00721 |

| Type | V9 | V10 | V11 | V12 | V13 | V14 | V15 | V16 | V17 |
|---|---|---|---|---|---|---|---|---|---|
| count | 284807 | 284807 | 284807 | 284807 | 284807 | 284807 | 284807 | 284807 | 284807 |
| mean | -2.41E-15 | 2.24E-15 | 1.67E-15 | -1.25E-15 | 8.19E-16 | 1.21E-15 | 4.89E-15 | 1.44E-15 | -3.77E-16 |
| std | 1.098632 | 1.08885 | 1.020713 | 0.999201 | 0.995274 | 0.958596 | 0.915316 | 0.876253 | 0.849337 |
| min | -13.4341 | -24.5883 | -4.79747 | -18.6837 | -5.79188 | -19.2143 | -4.49894 | -14.1299 | -25.1628 |
| 25% | -0.6431 | -0.53543 | -0.76249 | -0.40557 | -0.64854 | -0.42557 | -0.58288 | -0.46804 | -0.48375 |
| 50% | -0.05143 | -0.09292 | -0.03276 | 0.140033 | -0.01357 | 0.050601 | 0.048072 | 0.066413 | -0.06568 |
| 75% | 0.597139 | 0.453923 | 0.739593 | 0.618238 | 0.662505 | 0.49315 | 0.648821 | 0.523296 | 0.399675 |
| max | 15.59499 | 23.74514 | 12.01891 | 7.848392 | 7.126883 | 10.52677 | 8.877742 | 17.31511 | 9.253526 |

| Type | V18 | V19 | V20 | V21 | V22 | V23 | V24 | V25 | V26 |
|---|---|---|---|---|---|---|---|---|---|
| count | 284807 | 284807 | 284807 | 284807 | 284807 | 284807 | 284807 | 284807 | 284807 |
| mean | 9.56E-16 | 1.04E-15 | 6.41E-16 | 1.65E-16 | -3.57E-16 | 2.58E-16 | 4.47E-15 | 5.34E-16 | 1.68E-15 |
| std | 0.838176 | 0.814041 | 0.770925 | 0.734524 | 0.725702 | 0.62446 | 0.605647 | 0.521278 | 0.482227 |
| min | -9.49875 | -7.21353 | -54.4977 | -34.8304 | -10.9331 | -44.8077 | -2.83663 | -10.2954 | -2.60455 |
| 25% | -0.49885 | -0.4563 | -0.21172 | -0.22839 | -0.54235 | -0.16185 | -0.35459 | -0.31715 | -0.32698 |
| 50% | -0.00364 | 0.003735 | -0.06248 | -0.02945 | 0.006782 | -0.01119 | 0.040976 | 0.016594 | -0.05214 |
| 75% | 0.500807 | 0.458949 | 0.133041 | 0.186377 | 0.528554 | 0.147642 | 0.439527 | 0.350716 | 0.240952 |
| max | 5.041069 | 5.591971 | 39.4209 | 27.20284 | 10.50309 | 22.52841 | 4.584549 | 7.519589 | 3.517346 |

| Type | V27 | V28 | Amount | Class |
|---|---|---|---|---|
| count | 284807 | 284807 | 284807 | 284807 |
| mean | -3.66E-16 | -1.23E-16 | 88.34962 | 0.001727 |
| std | 0.403632 | 0.330083 | 250.1201 | 0.041527 |
| min | -22.5657 | -15.4301 | 0 | 0 |
| 25% | -0.07084 | -0.05296 | 5.6 | 0 |
| 50% | 0.001342 | 0.011244 | 22 | 0 |
| 75% | 0.091045 | 0.07828 | 77.165 | 0 |
| max | 31.6122 | 33.84781 | 25691.16 | 1 |

**Table 1 - 4 Represent different class comparisons for PAC features**

## Correlation evaluation

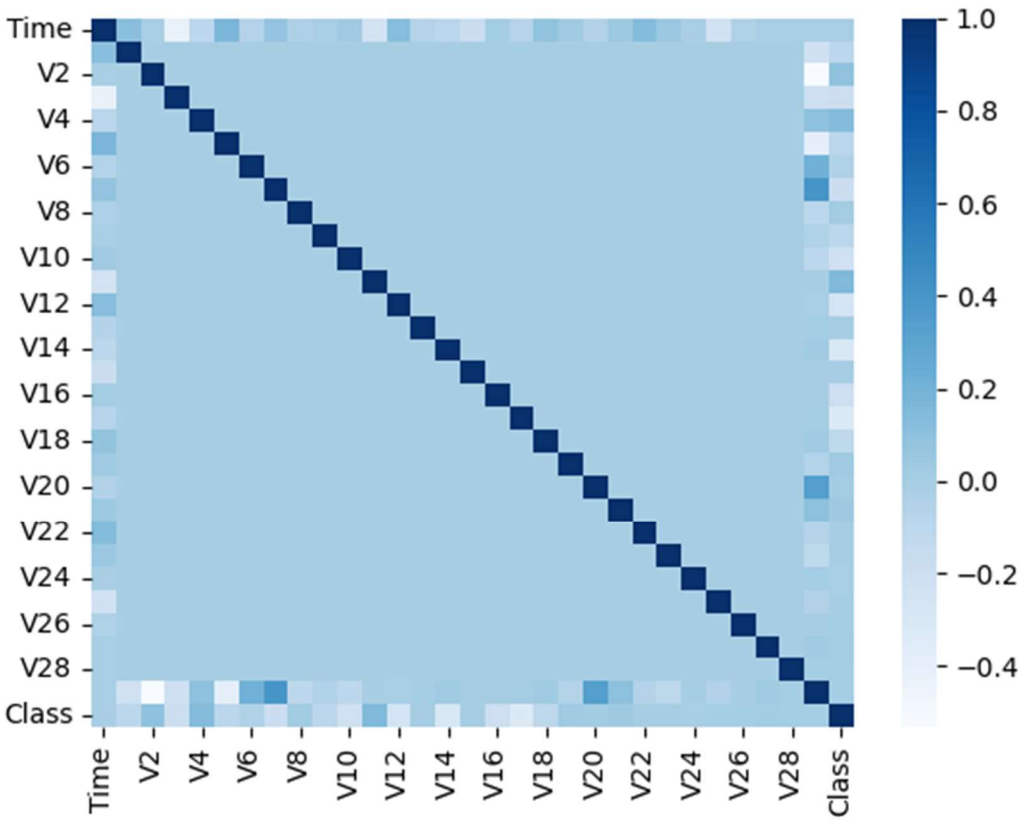


**Figure 1 Coerrelation evaluation**

The provided dataset has a very low correlation between the columns. Therefore, training models will lead to will not capture any crucial information.

For our training model, we have to remove the time of the transactions and amount as it made significantly less information to the output class that is determined.

# Implementation Details

## VAE Implementation

The VAE is implemented in models.py. There are two classes Generator and Discriminator. The generator has four linear hidden layers—two layers for feature extraction, one for mu calculation, and another for sigma. The decoder has three layers: the hidden layer for feature extraction and one layer for final output. The encoder takes an input feature and passes through the two layers then the output features are passed through the mu and sigma layers. Finally, these two layers are used to calculate the loss of the network.

## Logistic Regression Implementation

We have used a simple implementation of Logistic regression provided by scikit-learn. The loss equation is given by

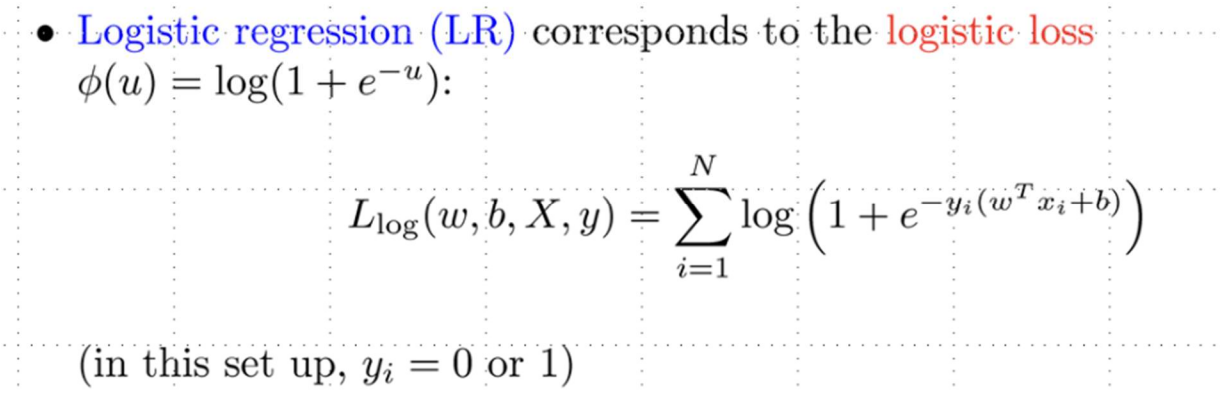


**Figure 2 Logistic Regression Formula**

# Experimental Results

## Baseline metrics

We evaluated the baseline metrics using Sci-kit learn logistic regression. We achieved a max accuracy of 63% over multiple runs and iterations. Testing only fraudulent transiting, we are gaining an accuracy of 60%. This looks like the model is overfitted to the data.

## Blending Data

We introduced 50 new records, generated using the VAE and reran the Logistic regression. We could see that LR models still overfitted the accuracy of the fraudulent significantly up to 80%.

## Result Graphs

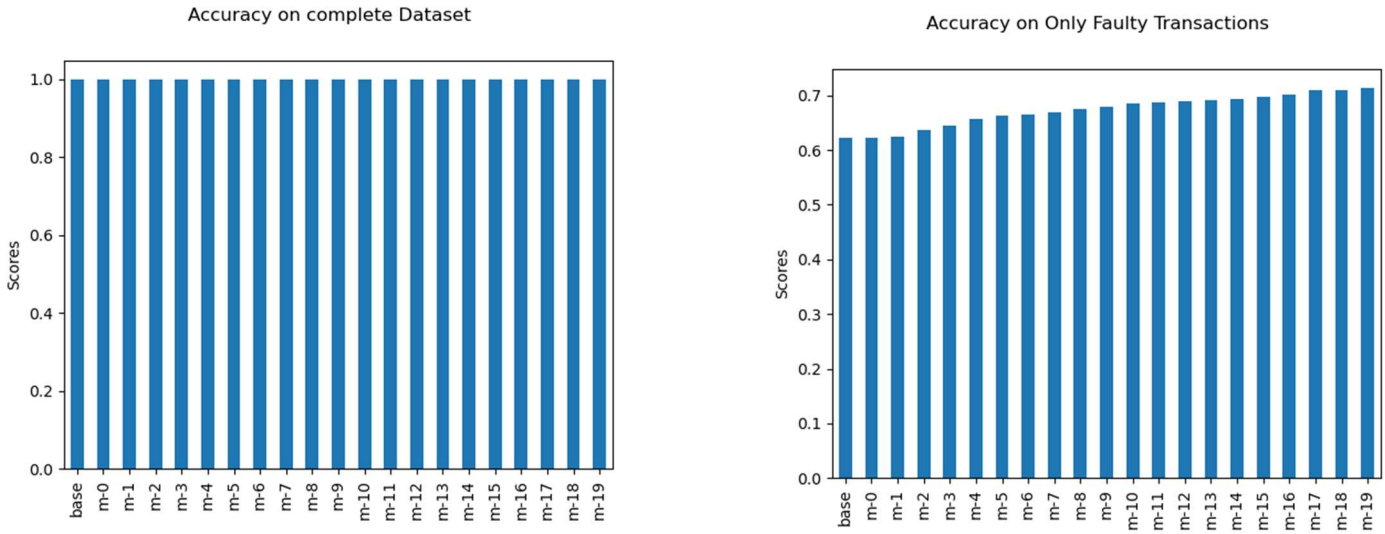


**Figure 3.**

**Accuracy on the complete test set (left), Checking the accuracy of on only fraudulent dataset (right)**

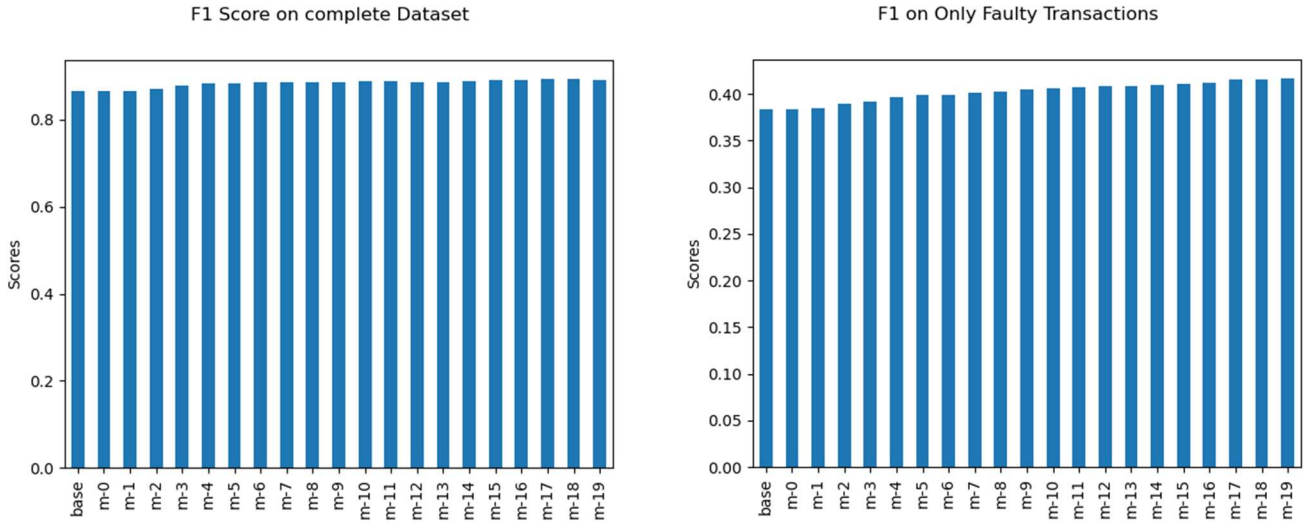


**Figure 4.**

**F – 1 scores on all dataset (left) and F1 – score on fraudulent dataset (right)**

## Conclusion

By adding new records generated using VAE, we were able to get the chances of pick up on fraudulent transactions. Having another model like XGboost or a neural network would significantly affect the accuracy. It also opens new possibilities to generating synthetic datasets that are similar to those provided.

## Further Work

By adding new data randomly generated via the torch.rand function in the model introduces many variances that create the model. Therefore, controlling and fine-tuning this parameter will allow the user to achieve at least 85% accuracy.

Blending the output of multiple models might yield better latent space results, reducing the current overfitting on the model.

## Project Repository

Github:- https://github.com/Tejasvedagiri/MS_DM_Term_Project.git

Dataset:- https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

## References


[1]Auto-Encoding Variational Bayes Diederik P. Kingma Max Welling Machine Learning Group Universiteit van Amsterdam https://arxiv.org/pdf/1312.6114.pdf .

[2] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In Parallel Distributed Processing. Vol 1: Foundations. MIT Press, Cambridge, MA, 1986.

[3] E. Oja. Simplified neuron model as a principal component analyzer. Journal of mathematical biology, 15(3):267–273, 1982. Autoencoders Dor Bank, Noam Koenigstein, Raja Giryes https://arxiv.org/pdf/2003.05991.pdf.

[4] G.E. Hinton and R.R. Salakhutdinov. Reducing the dimensionality of data with neural networks. Science, 313(5786):504, 2006.

[5] G.E. Hinton, S. Osindero, and Y.W. Teh. A fast learning algorithm for deep belief nets. Neural Computation, 18(7):1527–1554, 2006.

[6] Yoshua Bengio and Yann LeCun. Scaling learning algorithms towards AI. In L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors, Large-Scale Kernel Machines. MIT Press, 2007.

[7] https://medium.com/ai%C2%B3-theory-practice-business/understanding-autoencoders-part-i-116ed2272d35.

[8] G. E. Hinton* and R. R. Salakhutdinov Reducing the Dimensionality of Data with Neural Networks .

[9] Diederik P. Kingma Max Welling Machine Learning Group Universiteit van Amsterdam Auto-Encoding Variational Bayes.