Problem statement

A generic fraud detection framework that works irrespective of how many frauds are available in actual data and can be retrained if new modus operandi of frauds or anomalies arise.

The framework has been tested on detecting:

1. **Money mules**: This use case focuses on testing transaction data related frauds or anomalies
2. **Sourcing frauds**: This use case focuses on testing the framework on detecting fraud patterns in non-transaction data where frequency of fraud is 1 in 10000 cases.

In both the cases we're trying to maximize recall keeping it closer to one while keeping precision more than 50%

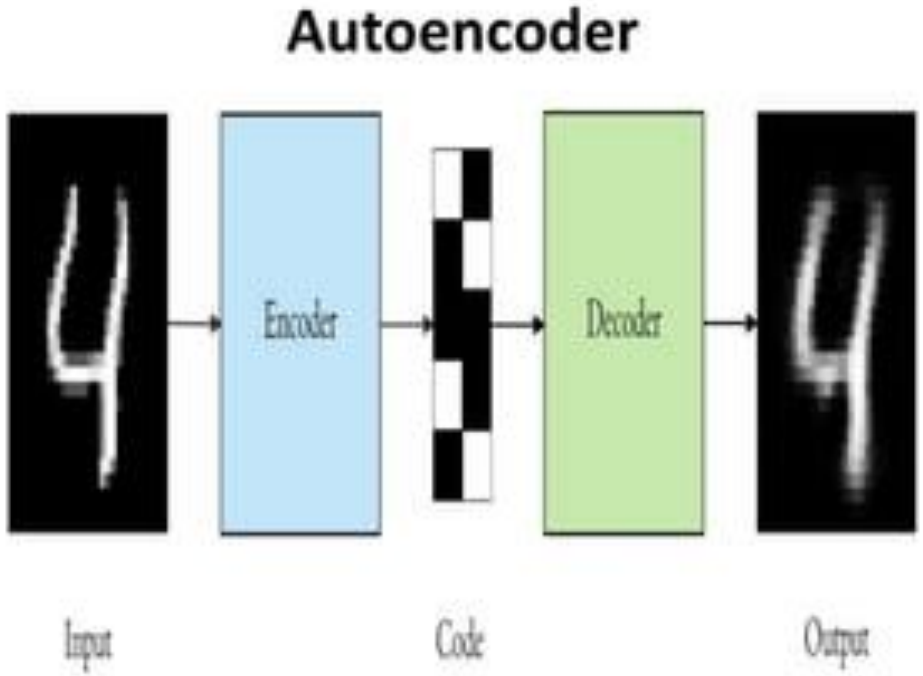
Need to develop a utility for HDFC Bank that would help them detect money mule accounts or can be used in any other fraud detection problem using semi-supervised deep learning techniques (autoencoders to be specific)

Autoencoders?

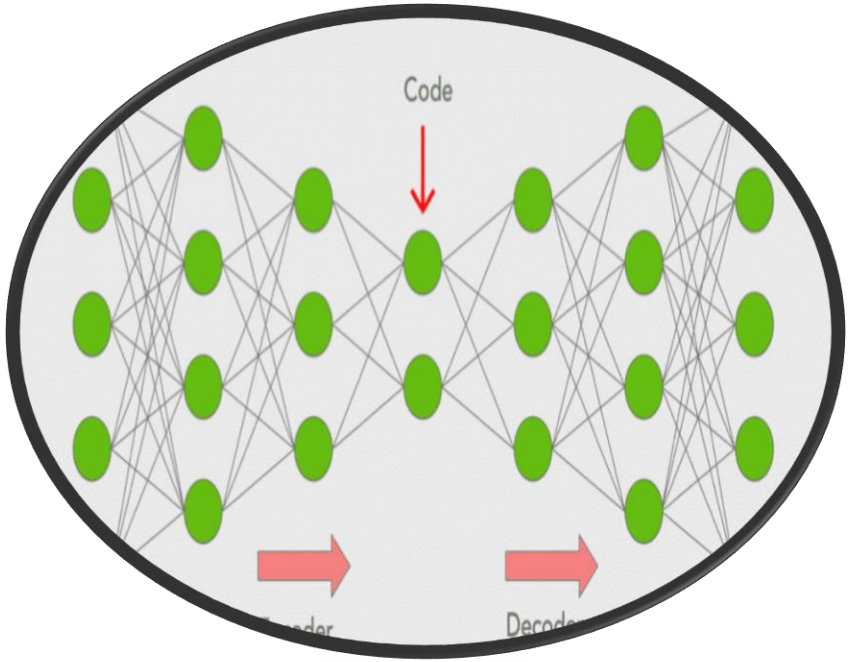
Autoencoders are a special type of feed forward **neural network** where input is the same as the output. They compress the input into a lower-dimensional code and then re-contruct the output from this representation. This code is a compact "summary" or "compression" of the input, also called the **latent-space representation.**

Why autoencoders?

1. It can handle even E-4 bad rate very well
2. The technique used earlier was flagging 13% people as frauds but this technique works way better
3. If new patterns of frauds are accomplished by tricksters, the pipeline can be retrained for new types of modus operandi. Pretrained autoencoders can be retrained to new modus operandi of frauds
4. Autoencoders don't require labelled data, saving costs associated with data annotation and labelling
5. It can investigate relationship between features that contributes to normal data vs. Identifying frauds without the need for much manual inspection



Full explanation of technique



Model algorithm | Data preprocessing | Feature selection

Note

1. Same autoencoder architecture is used everywhere
2. Number of layers of autoencoder is fixed
3. Output and bottleneck activations are fixed

Scaled dev, oos, oot

Training set = dev
Validation set = oos
Test set = oot

Training set, Validation set = frauds + non-frauds

frauds

Train autoencoder using frauds (all features)

Get descending list of features increasing RE

(Using SHAP or FPI method)

non-frauds

Train autoencoder using non-frauds (all features)

Get descending list of features increasing RE

(Using SHAP or FPI method)

Bottom features

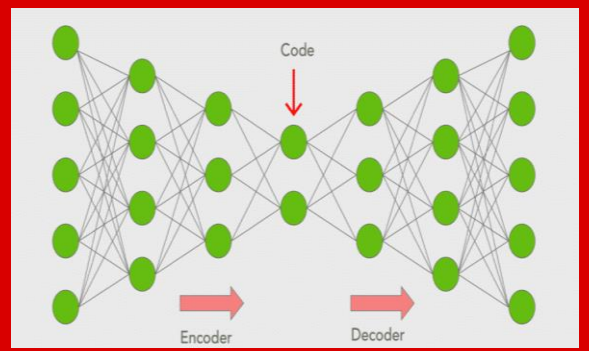
Top features

Drop these features from training & test set

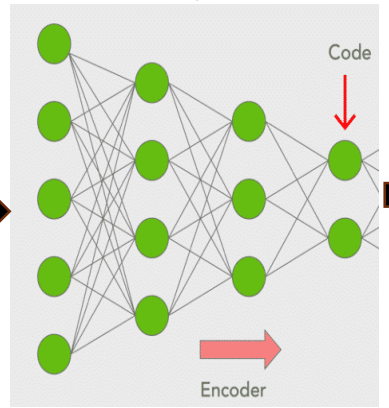
sets

frauds (or) non-frauds

Train autoencoder using Frauds (or) Non-frauds of training set



encoder



Encoded Train set
&
Encoded Test set

(We've kept SHAP explainability here)

Train & Test logistic regression model

Classification - Restricted

Hyperparameter module

Since it's an ensemble model. Hyperparameters tuning of the whole pipeline is done in 2 parts.

1. **Tuning autoencoder parameters**

   Autoencoder based

   - *train_on* - train autoencoder on frauds or non-frauds
   - *activation* - what activation function to use for hidden layers of autoencoder
   - *alpha* - if 'elu' activation chosen then what alpha to chose
   - *layer_size_0.8* - first layer of autoencoder should have what ratio of input features
   - *layer_size_0.5* - second layer of autoencoder should have what ratio of input features
   - *layer_size_0.2* - bottleneck layer of autoencoder should have what ratio of input features
   - *patience, min_delta, batch_size, epochs, val_split*

   Feature selection based

   - *feature_threshold* - what percentage of features to drop from list increasing RE of non-frauds

2. **Tuning logistic regression parameters** (after finding the best autoencoder parameters)
   - percentile_threshold - logistic threshold percentile
   - C - regularisation parameter
   - max_iter, etc.

Methods used for feature selection

1. **SHAP value method**
   - Calculates reconstruction error of all datapoints in a sample and selects the datapoint with maximum RE
   - Breaks down that datapoint's error into individual feature errors and ranks those features
   - Takes top 20% features and updates their weights to 0 input one at a time and calculates shap values of the autoencoder each time to get a much accurate contribution of that feature.
   - This method works with samples of data (1000) and takes the most time to run but is pretty accurate.
2. **FPI (feature permutation importance)**
   - Shuffles the values of each feature one at a time and observes the change in the reconstruction error for the autoencoder to get to know the importance of that feature
   - Takes all datapoints into account but takes 15 mins to run on 2 lakh data
3. **Direct reconstruction**
   - Simplest method out of the 3
   - Calculates reconstruction error of the while data and add all the datapoints column wise to get a general reconstruction error caused by each feature
   - Takes all datapoints into account and runs pretty fast but feature selection becomes hard here because of less variation in distribution

Feature selection intuition

There are 2 steps in dropping features

1. Features dropped using autoencoder trained on non-frauds

- Train autoencoder on non-fraud examples of training set taking all features
- Using SHAP or FPI method get a descending list of each feature contribution in increasing reconstruction error of non-fraud examples using the above autoencoder.
- Since we don't want features increasing RE of non-frauds (because they contribute to false positives)
- Looking at their distribution, we can decide what top features to drop from this list

2. Features dropped using autoencoder trained on frauds

- Train autoencoder on fraud examples of training set taking all features
- Using SHAP or FPI method get a descending list of each feature contribution in increasing reconstruction error of fraud examples using the above autoencoder.
- Since this list gives you features that increase the reconstruction error of frauds, hence helping to detect them as anomalies. We absolutely want these features.
- Looking at their distribution, we can decide how many features that do not contribute anything in re-constructing fraud examples should be dropped

(Plots excel summary to be shown)

POC results on OOT - SRC (123 features)

Using FPI method

1. Dropping top [redacted] of non-frauds list + dropping bottom [redacted] of frauds list
2. Logistic threshold - 99.9%
3. [redacted]% population triggered as fraud - [redacted] people
4. [redacted] out of [redacted] actual frauds were detected

	precision	recall	f1-score	support
0.0	1.00	1.00	1.00	[redacted]
1.0	0.25	0.68	0.36	[redacted]
accuracy			1.00	[redacted]
macro avg	0.62	0.84	0.68	[redacted]
weighted avg	1.00	1.00	1.00	[redacted]

[[5[redacted]0]
 [[redacted]3]]

Using SHAP method

1. dropping top [redacted] from non-frauds list + dropping bottom [redacted] from fraud list
2. Logistic threshold - 99.9%
3. [redacted]% population triggered as fraud - [redacted]
4. [redacted] out of [redacted] actual frauds were detected

	precision	recall	f1-score	support
0.0	1.00	1.00	1.00	5[redacted]9
1.0	0.30	0.84	0.44	[redacted]9
accuracy			1.00	5[redacted]3
macro avg	0.65	0.92	0.72	5[redacted]3
weighted avg	1.00	1.00	1.00	5[redacted]3

[[5[redacted]]
 [[redacted]]]

POC results on DEV+OOS - SRC (123 features)

Using FPI method

1. Dropping top [redacted] of non-frauds list + dropping bottom [redacted] of frauds list
2. Logistic threshold - 99.9%
3. [redacted] % population triggered as fraud - [redacted] people
4. [redacted] out of [redacted] actual frauds were detected

	precision	recall	f1-score	support
0.0	1.00	1.00	1.00	1[redacted]3
1.0	0.29	0.79	0.43	[redacted]8
accuracy			1.00	1[redacted]1
macro avg	0.65	0.90	0.71	1[redacted]1
weighted avg	1.00	1.00	1.00	1[redacted]1

[[1[redacted]]
[[redacted]]]

Using SHAP method

1. dropping top [redacted] from non-frauds list + dropping bottom [redacted] from fraud list
2. Logistic threshold - 99.9%
3. [redacted] % population triggered as fraud - [redacted] people
4. [redacted] out of [redacted] actual frauds were detected

	precision	recall	f1-score	support
0.0	1.00	1.00	1.00	1[redacted]3
1.0	0.29	0.79	0.43	[redacted]8
accuracy			1.00	1[redacted]1
macro avg	0.65	0.90	0.71	1[redacted]1
weighted avg	1.00	1.00	1.00	1[redacted]1

[[1[redacted]]
[[redacted]]]

POC results - Money mule (273 features)

Using SHAP method

1. dropping top [redacted] from non-frauds list + dropping bottom [redacted] from fraud list
2. Logistic threshold - 97.91%
3. [redacted]% population triggered as fraud - [redacted] out of which [redacted] were actual frauds
4. [redacted] out of [redacted] actual frauds were detected

	precision	recall	f1-score	support
0.0	0.99	0.99	0.99	[redacted]
1.0	0.56	0.56	0.56	[redacted]
accuracy			0.98	[redacted]
macro avg	0.77	0.77	0.77	[redacted]
weighted avg	0.98	0.98	0.98	[redacted]

[[[redacted]]
 [[redacted]]]