2023

Insurance Fault Detection Using Unsupervised Sequential Anomaly Detection

TABLE OF CONTENT

1. Metrices, Trade-off and Module Evaluation

1.1. Describe The Metrics

Detection rate is the number of claims detected as fraud to be investigated, to the total number of insurance claims. Hit rate is the number of cases where payment is rejected on a fraudulent claim to the number of investigated claims.

1.2. The Trade-off

To enhance the fraud detection rate, a model which would detect more suspicious cases is needed. However, this may result in an increase in false positives, hence a decrease in the hit rate, and consequently to an increase in expenses incurred on investigating non-fraudulent cases.

On the other hand, to improve the hit rate, a more accurate model that flags fewer cases as suspicious is needed. However, this approach may potentially result in some cases of actual fraud remaining undetected.

Despite the aforementioned drawback, we believe increasing the hit rate holds greater significance, as it limits unintended bias. Augmenting the detection rate, and hence the quantity of false positive fraud claims, introduces discriminatory bias towards individuals who for example may utilize lower-resolution cameras. Therefore, a more precise model enhances both the hit rate and restricts partiality within the algorithm.

1.3. Model Evaluation

Fraud that remains undetected (false negatives) negatively affects honest policyholders, as they are faced with increased premiums which are calculated based on the collective pooled risk. Hence, increasing the detection rate of suspicious claims may protect these policyholders.

A model that detects more fraud however, may be costly for the insurer. Suspecting a higher number of fraudulent claims means that fraud detection teams may spend time and money investigating false positives instead of actual fraud cases. Streamlining its fraud detection operations would also mitigate increased premiums for policyholders in the long-run.

Therefore, we recommend that Shift prioritize improving the hit rate of its fraud detection algorithm. This approach will be sustainable and rewarding for both the insurer company and the policyholders.

2. The Simple Model

The model provides the detection and hit rate for different values of *t*. The hit rate is the proportion of true positive cases (actual frauds that are correctly classified as frauds) out of all detected frauds. Further, the detection rate is the proportion of true positive cases and false negative cases (all classified as frauds) out of all cases in the dataset. Choosing a value for t depends on the trade-off between the hit-rate and the detection rate.

The logic underlying this model is to classify claims as fraudulent if the time difference between the loss date and the first policy subscription date is less than or equal to a threshold value, t. The idea behind this is that fraudulent claims may be more likely to be made soon after a policy is taken out, while legitimate claims are more likely to occur later.

The results of the simple model suggest that a sensible value of "t" is 10 days (where hit rate is at 19% and detection rate is at 3%). By defining t=10, the company should flag incidents as suspicious if the claim happened within 10 days from subscription date to the loss date. This can be also inferred through the plots of the hit and detection rate against different values of "t", where hit rate first increases for very small values of “t” and then keeps decreasing (appendix A). We have selected a threshold of 10 days by prioritizing a high hit rate, and simultaneously considering not to have a really low detection rate.

The simple model is an easy and transparent method to classify claims as fraud or not fraud. However, the model assumes that all claims occurring within “t” days of subscription date are suspicious (only considering a single regressor). Therefore, the model does not take into account other features that could make a claim more or less likely to be suspicious.

3. Decision Tree Classifier

As an intermediate model, we fit a decision tree classifier to predict whether claims were fraudulent or not. We use the validation set to identify the optimum number of leaves to create the "best tree". To determine the ideal classification threshold, we utilize the J statistic (also known as the Youden Index) based on the ROC curve. This metric is calculated by maximizing the sum of sensitivity and specificity subtracted by 1, where sensitivity refers to the true positive rate and the specificity indicates the true negative rate. When applying this threshold, we found that the detection rate to be approximately 1.2% and 20 % respectively (appendix B).

Although the model seems to be performing quite well when evaluated by the AUC score which is around 0.83, this is significantly affected by the imbalanced nature of the classes in the training data. Decision tree classifiers are prone to biased learning when there are more instances of one class than another, resulting in higher accuracy rates for the majority class and a lower for the minority class. In this case, the decision tree classifier we built seems to be biased towards non-fraudulent claims which make up 99% of the data, but it fails to accurately detect fraudulent cases as they are rare instances in the training data.

4. Neural Network

For our complex model, we are building a neural network using TensorFlow, in order to predict claims as "fraudulent" or "not fraudulent". We started from a Stochastic Gradient Descent (SGD) neural network of 2 hidden layers of 10 units each with tuning learning rate at 0.01 and momentum at 0.9. However, the accuracy did not improve after reaching 99%. To enhance the accuracy, we experimented with multiple several modifications, such as; lowering the learning rate to 0.001, increasing the number of units per hidden layer to 20, adding L2 regularization and incorporating an extra hidden layer with a batch size of 16. However, none could surpass a 99% training and validation accuracy.

Furthermore, we attempted to create a model with RMSprop with two hidden layers, 20 units each, 0.02 dropout rate, L2 regularization and a batch size of 16. However, the training and validation accuracy did not improve beyond 99%.

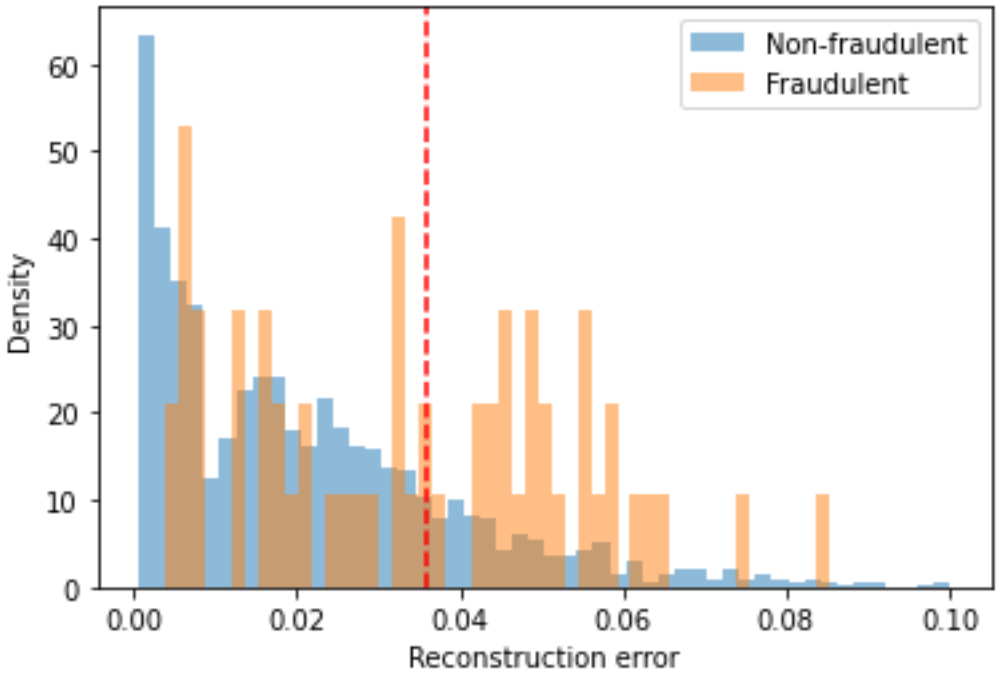
Next, we have attempted to create a neural network using ADAM algorithm, as it combines the advantages of gradient descent and RMSprop, resulting in 98.99% training accuracy, 99.00% validation accuracy.

Finally, the final neural network has 92 hidden units, batch size of 32, learning rate of 0.0003 and a dropout rate of 0.14, but its accuracy remained fixed at 99%. The predicted fraud probabilities on the test data are typically close to 0. Furthermore, the detection rate drops greatly as the threshold increases, while the hit rate increases and maximizes at 18% at 12% threshold (appendix C).

5. Autoencoder

For the final model, we have firstly created a training dataset containing only non-fraudulent claims and a validation and test set containing both fraudulent and non-fraudulent claims. We have then used TensorFlow to build an autoencoder. We have tried different ways of tuning parameters (which can be found in the submitted csv file) and the chosen model works as follows: we first scaled the dataset, and then built an encoder and decoder, both made up of 3 neuron layers, using tanh and RElu activation functions. Number of Epochs used is 10 and a batch size of 32. To improve the results of the autoencoder, we have also used checkpoint and early stopping.

Additionally, we plotted a graph illustrating the changes in training and validation loss throughout the training process of the model. Our results revealed that as the number of Epochs increases, both training and validation loss decrease, indicating that the autoencoder is learning and improving its predictive performance (appendix D).



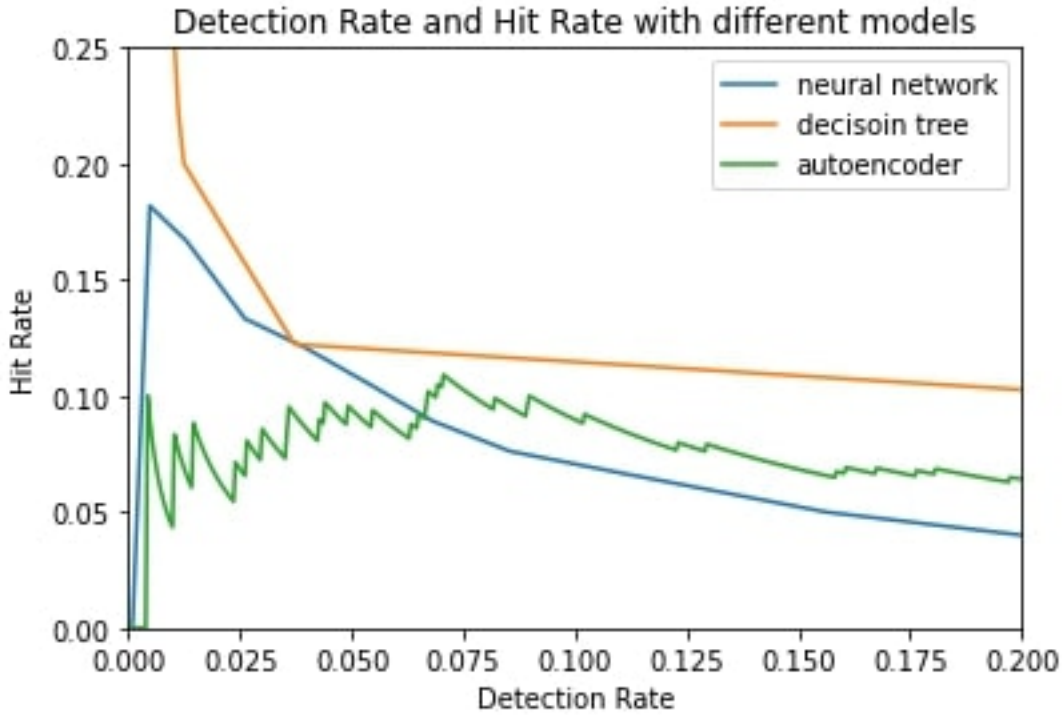
The performance of an autoencoder can be evaluated using reconstruction error, which measures the difference between the input data and the reconstructed output. Ideally, there should be a distinguishable distinction between the two classes. Based on this histogram, we are choosing a threshold of 0.036. For the selected threshold, the hit rate is around 7% and detection rate is around 17% (appendix E).

6. Different Approaches

6.2. Hit & Detection Rate

We evaluated the hit and detection rates for various models at different thresholds (appendix F). It is noteworthy that the decision tree model outperforms the autoencoder and neural network in terms of hit rates, regardless of the level of detection rate.

At the same detection rate, the neural network performs the poorest with a 9% hit rate. This outcome could be attributed to the model's tendency to overfit fraudulent instances to reduce the Gini Index more efficiently. In contrast, the neural network' ability to learn is hindered by the insufficiency of fraudulent observations.



6.3. The Transparency Behind the Approaches

The simple model is theoretically considered transparent as it only considers one variable. However, its simplicity is undoubtedly a huge drawback in the accuracy of predictions as it ignores all other features in the classification process.

Although decision trees are generally considered transparent due to their intuitive structure, our fraud dataset involves more than 70 unique features which causes the tree to become more complex, deeper, and thus more difficult to interpret.

Our neural network is a highly complex model and is made up of many hidden layers, and non-linear activation functions, making it difficult to interpret how the model makes predictions.

The autoencoder is less transparent than both the decision tree and the neural network as it is used as a black-box with limited interpretability. This underlying complexity makes it difficult to understand what specific features the model is using to learn a compressed representation of the actual data, without explicit influence from a target variable.

A lack of transparency can create issues as we cannot provide justification for fraudulent flagging by the model. This can lead to suspicions about potential bias in in the algorithm. Additionally, the investigation of suspicious claims would likely take more time without a clear indication of a target feature to investigate.

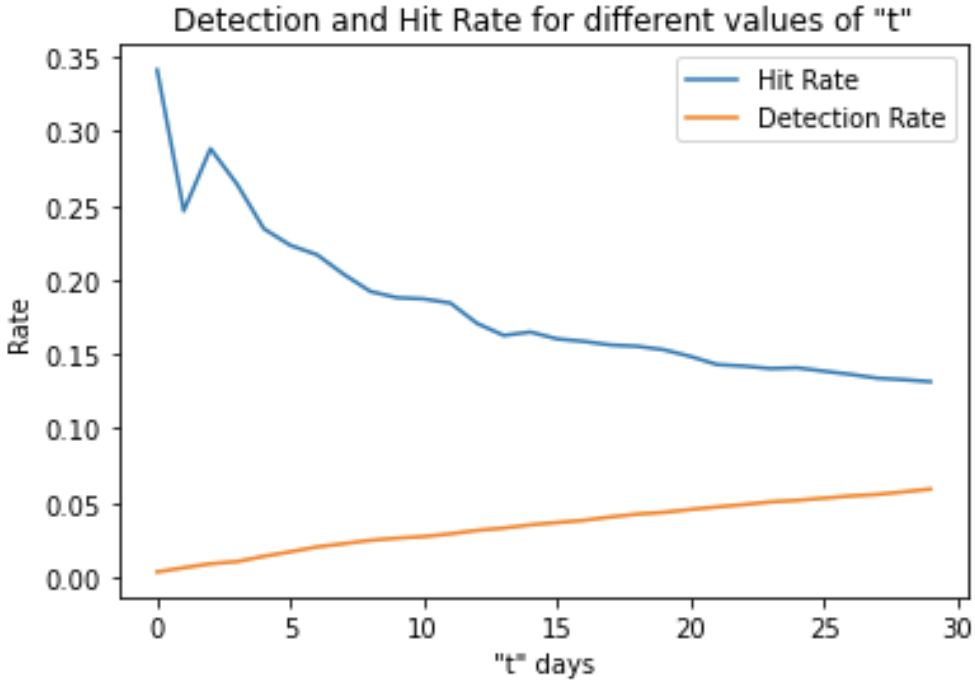
7. Improving Insurance Fraud Prediction

To mitigate the effects of imbalanced data on our models and achieve greater results, a useful approach could include random resampling methods. Given that our data set is predominantly biased towards non-fraudulent claims, we could under sample this class (removing observations) or oversample the minority class (adding more observations), i.e., the fraudulent claims. Nevertheless, these methods have their drawbacks. While undersampling is likely to lead to an inaccurate representation of the population, oversampling may cause overfitting, causing inaccurate prediction.

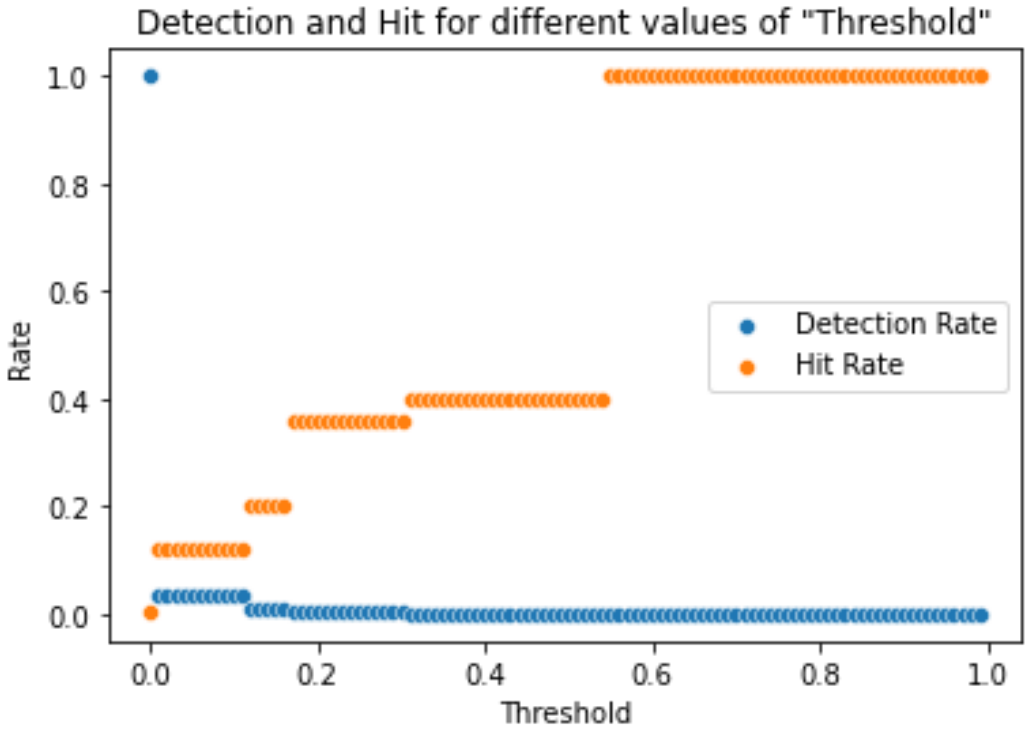
Additionally, for models such as neural networks, adding noise to them during their training phase can improve the robustness of the model and mitigate the effects of imbalanced data, resulting in both faster learning and more accurate generalization.

8. Appendices

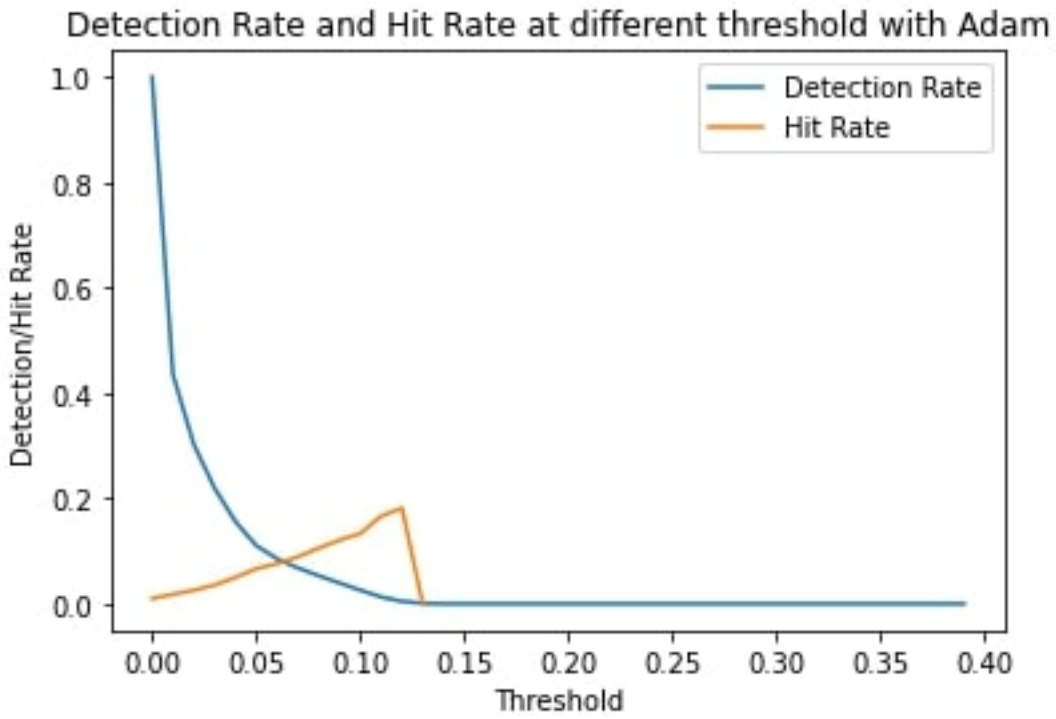
8.1. Appendix A



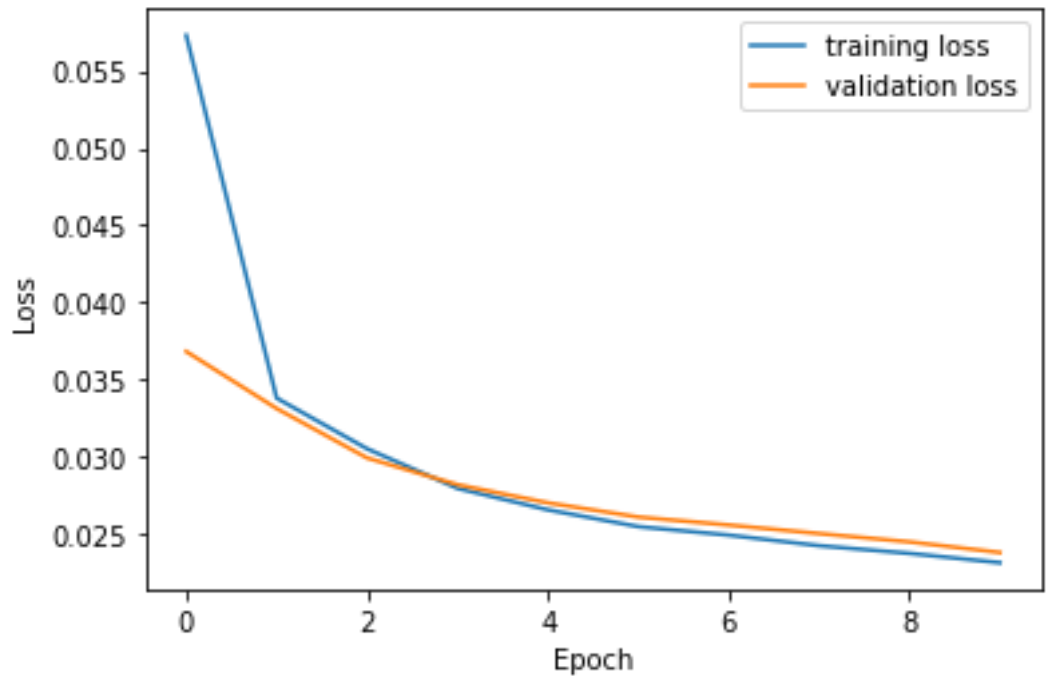
8.2. Appendix B



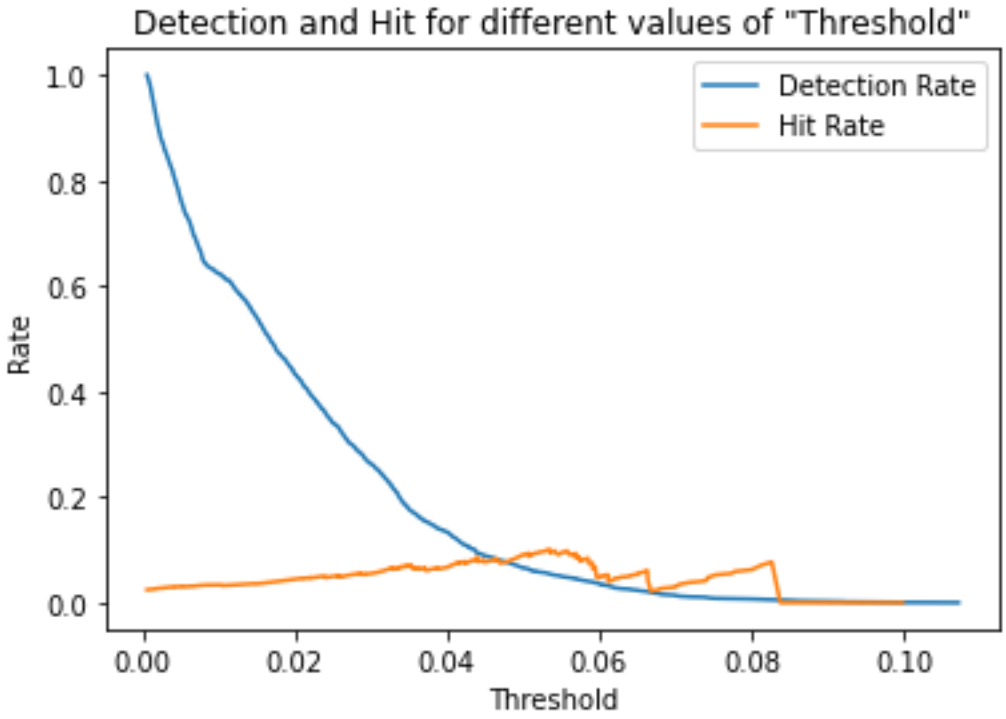
8.3. Appendix C



8.4. Appendix D



8.5. Appendix E



8.6. Appendix F

	Simple Model	**Decision Tree Classifier**	**Neural Network**	**Autoencoder**
Hit Rate	10%	20%	9%	7%