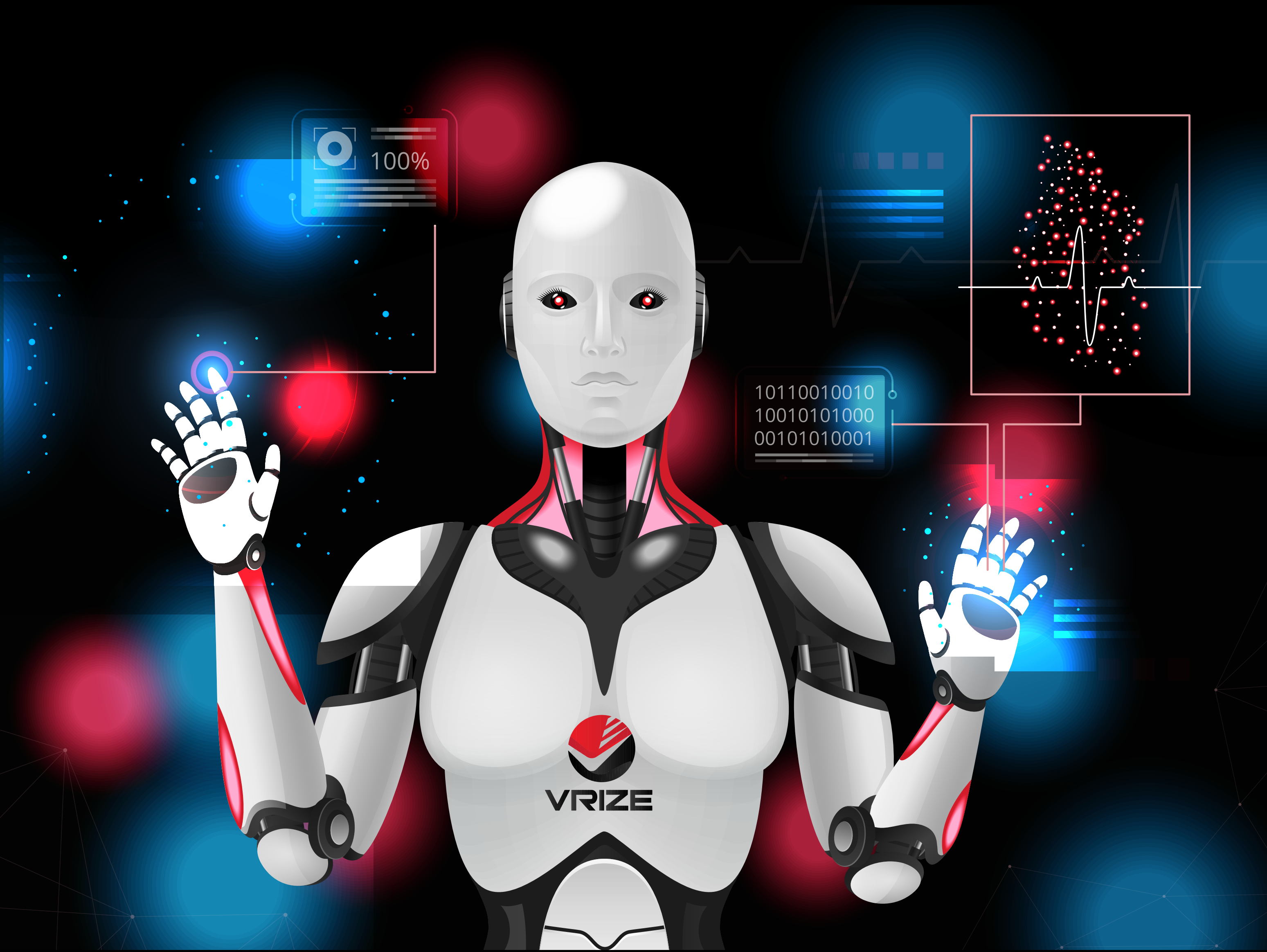
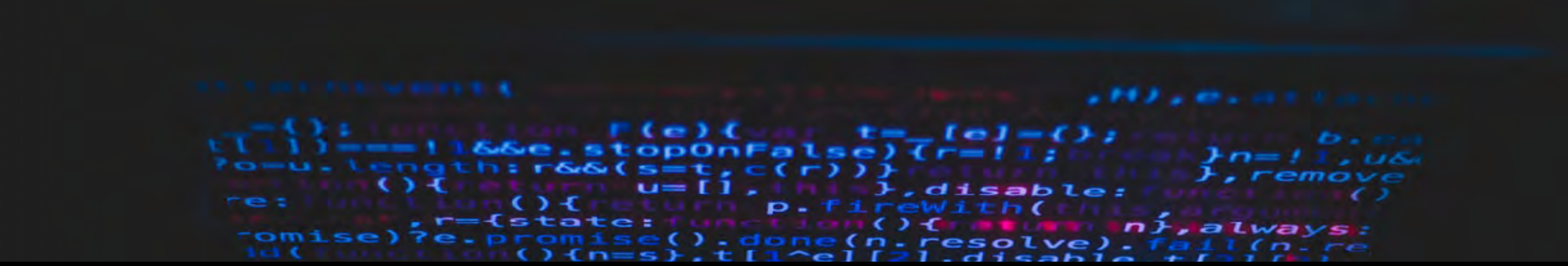
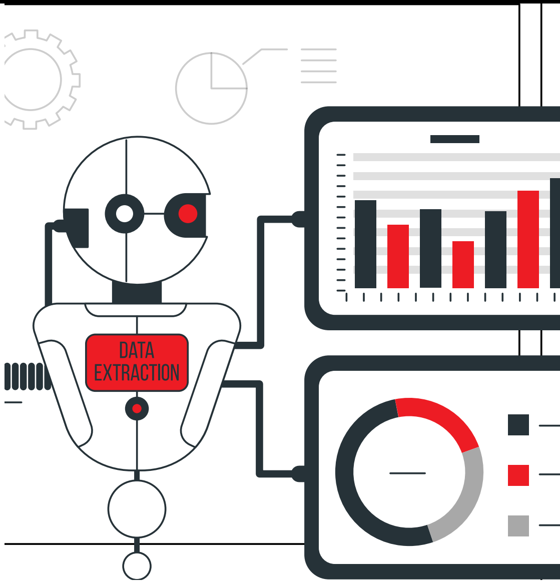
A Meta-learning Approach To Validate Machine Learning Models With Unlabeled Data





Introduction

With the explosion of data generation (based on a potent combination of internet accessibility, and proliferation of mobile devices), and accessible compute power, the use of machine learning models to solve issues has also seen progress at a comparable pace. While the usage of models has existed for decades, what has been novel and exciting is the deployment of models which use so called human-in-the-loop machine learning. A classic example of this is an image recognition model, such as AlexNet.



A key identifying feature of human-in-the-loop machine learning models is that they necessarily require human involvement for both, model creation, and assessment of its performance. Typically, human involvement is needed to label the data so that supervised learning algorithms can be used with the data. In the AlexNet example, the image recognition model could be trained only when humans annotated the ImageNet data with labels and bounding boxes (fun fact: CAPTCHA systems are a publicly outsourced implementation of human annotation).

While annotating the data in the launch cycle necessarily requires a human-in-the-loop (at least for now), is it really necessary to need a similar human-in-the-loop involved during assessment of model performance for later cycles?

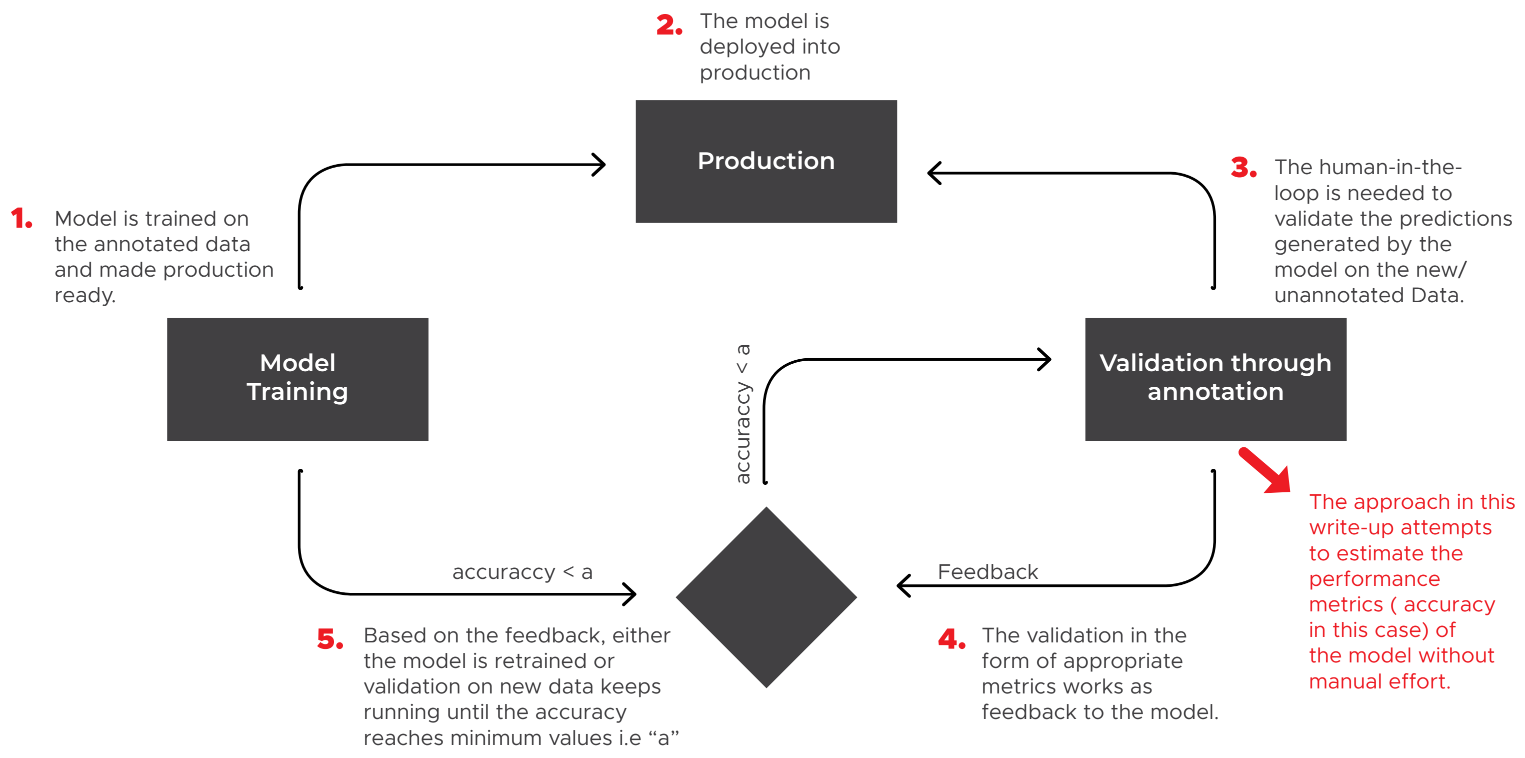

2. The model is deployed into production
Production
3. The human-in-the-loop is needed to validate the predictions generated by the model on the new/ unannotated Data.
1. Model is trained on the annotated data and made production ready.
Model Training
accuraccy < a
Validation through annotation
The approach in this write-up attempts to estimate the performance metrics (accuracy in this case) of the model without manual effort.
accuraccy < a
Feedback
5. Based on the feedback, either the model is retrained or validation on new data keeps running until the accuracy reaches minimum values i.e "a"
4. The validation in the form of appropriate metrics works as feedback to the model.

Figure 1: ML model in production flow chart.

Based on research in this area, we propose a methodology to address this problem referred to as meta-learning, and also provide our insights on the scope for generalized application.

Literature Review & Approach Summary

The research paper cited for this write-up is "Estimating Accuracy for Text Classification Tasks on Large Unlabeled Data" by Chaturvedi, Faruquie, Subramaniam, Mohania (2010) (linked here). While the paper proposes the approach within the context of text data and further tests on similar data, the last section of our write-up will include our insights (including code snippets) on how well it generalizes to other problems / data / metrics.

The gist of the approach, as detailed in the paper, is as follows: the main model is trained leveraging the human-in-the-loop annotation done at the start on the full data. From this data, various samples are generated. The accuracy (or any relevant metric) of the model is computed on said sample. Further, meta-features (will explain this shortly) are generated for this sample. This process is repeated across a sufficient number of samples, with the metric value and meta-features being recorded each time so that they now form their own dataset. A separate model is then developed on this dataset. The resulting model essentially predicts the accuracy of a model on any data based on the meta-features of said data (hence, this approach is referred to as meta-learning). This model can then be used to predict the accuracy of the original model on a new un-annotated data.

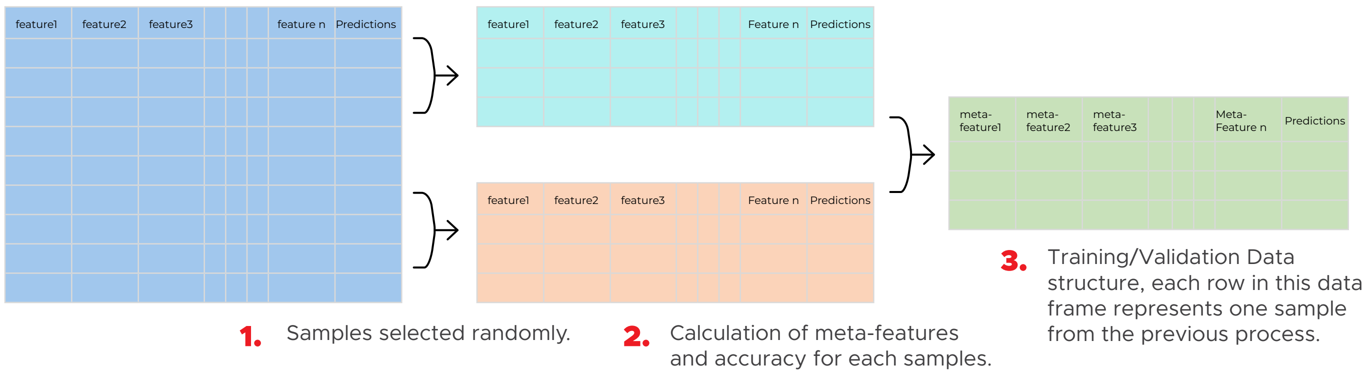

feature1
feature2
feature3
feature n
Predictions
feature1
feature2
feature3
Feature n
Predictions
feature1
feature2
feature3
Feature n
Predictions
meta-feature1
meta-feature2
meta-feature3
Meta-Feature n
Predictions
1. Samples selected randomly.
2. Calculation of meta-features and accuracy for each samples.
3. Training/Validation Data structure, each row in this data frame represents one sample from the previous process.

Figure 2: Sampling method for creation of Training/Validation Data.

Sampling & Meta-Features

Since the dataset basis which the meta-learning model will be trained is dependent on the sampling scheme of the original dataset (on which the main model was trained using annotations), a few principles will help detail the scheme requirements. These principles are, broadly, as follows.

1. The number of unique (but not necessarily distinct) samples should be sufficient for the robust estimation of the meta-learning model. We recommend a minimum of 30 samples, so that the corresponding 30 data points (accuracies and meta-features) can be used to develop the meta-learning model.

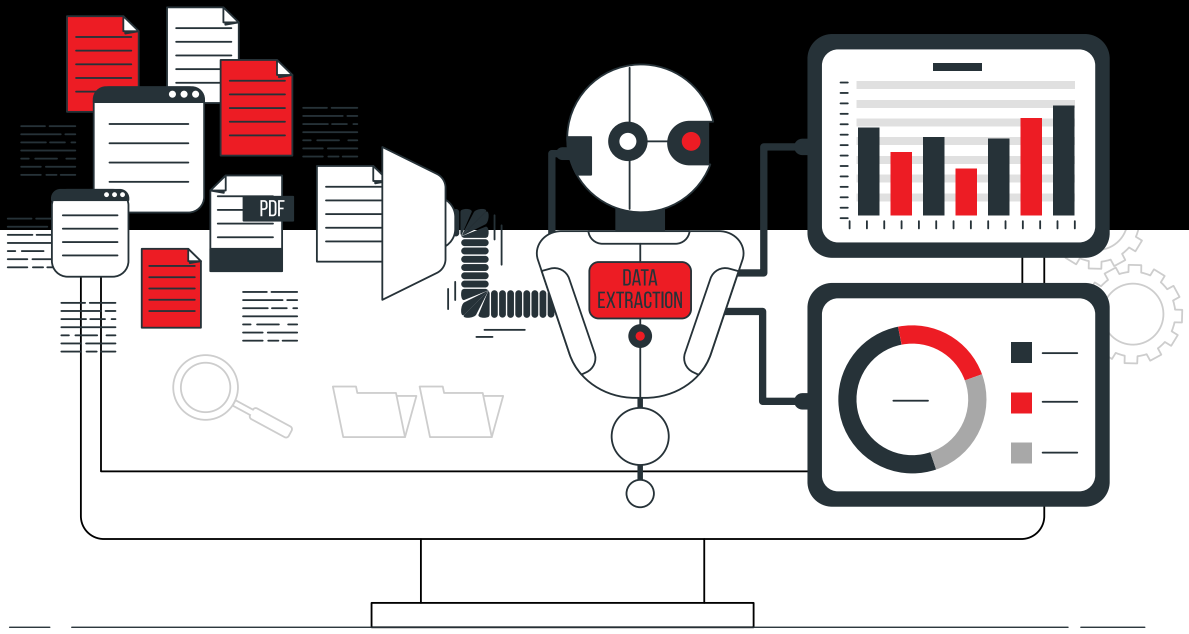
2. The previous point implies that if k-fold validation like sampling scheme is being used, the value of k should be at least 30. Remember, using k-fold validation necessarily implies that the samples are not just unique, but also distinct.

3. Irrespective of whether samples are generated through a k-fold mechanism, or some other mechanism (where the samples are not necessarily distinct), simple random sampling without replacement is always a good default option. If there is a strong set of candidates for stratification, stratified sampling can be used as well..

The next key components are the meta-features. Meta-features are a way to capture the characteristics (not the values themselves) of the data basis which the model was trained and a particular accuracy achieved. The meta-features of unlabeled data (say of data used by the model in production) can then be used to predict how the model accuracy may materialize, given the relationship learned (through the meta-learning model) between the two across the various models trained on various samples.

Meta-features can be of two types: based on the features of the models themselves, or based on the predicted outputs of the models.

Let's try and outline this with the help of an example. Suppose that 30 samples are generated as per the principles listed. For each sample, consider the columns which drive the prediction of the model, say 10 features (F1, F2, F3,...,F10). For each sample, the meta-features for a single feature (say, F1) can be defined as follows: any measure that characterizes the distribution of said feature in that specific sample is a meta-feature. For example, for F1, the mean, standard deviation, skewness, kurtosis etc. are examples of meta-features. These are basic examples, and there is a lot of scope for creativity here. One could use distance based meta-features, such as Kolmogorov-Smirnov distance, Bhattacharya distance etc.



Meta-features based on the predicted outputs of the model follow a similar approach. Suppose that the output of the model is a probability denoting the matching-ness of two images. For each sample, there will be a set of predicted probabilities, which can be meta-featurized by considering the same measures: mean, standard deviation, skewness, kurtosis etc. Accordingly, values will be generated for each sample on which a model has been trained.

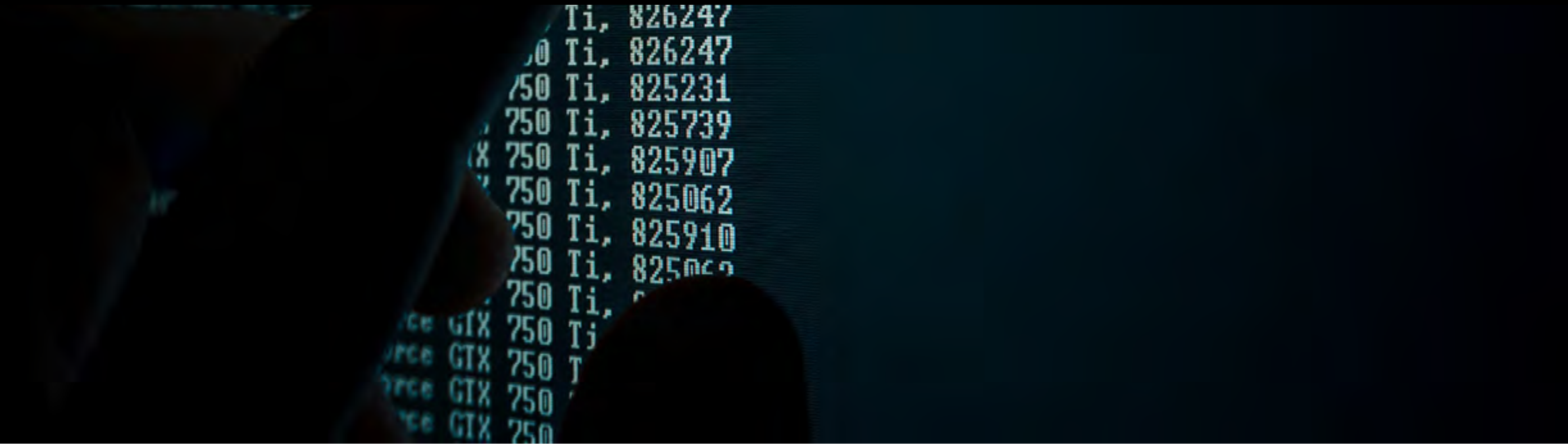
Since the value of these meta-features will change based on the sample, the full tabular representation of the data will look as follows. This is the data on which the meta-learning model will be trained. Once trained, the accuracy of the model on new data with unlabeled data can be predicted by calculating the same meta-features of the new data and applying the meta-learning model.

Sample	F1 MF1 (mean)	F1 MF2 (standard deviation)	Predicted Output MF1 (mean)	F10 MF3 (skewness)	F10 MF4 (kurtosis)	Accuracy
1						
2						
.						
.						
.						
.						
30						

Generalization and Code Snippet

Hopefully, the simplicity and power of this meta-learning approach has impressed you. We would like to conclude this write-up by remarking on the generalization of this approach.

While the original paper tested this approach in the context of a text classification task with accuracy as the metric of interest, our own experimentation has revealed that it generalizes well when used with other problem statements and even other metrics of interest.



As a closing bonus, the following code snippet / notebook records our attempt at using this approach with the publicly available Emotion prediction data, and Math Lectures Data on Kaggle. As you can see, the approach predicts the accuracies with RMSE loss close to zero. You can also validate from the Evaluation Graph at the end. The Notebook with explanations can be found here: Meta-Learning-Notebook

Contact Us
Info@vrize.com