Smart Digital Forensic Framework for Crime Analysis and Prediction using AutoML

Sajith A Johnson[1]
M. Tech, Department of Computer Science and Engineering (Cyber Security and Digital Forensics), Koneru Lakshmaiah Education Foundation, Vaddeswaram - 522502, Guntur Andhra Pradesh, India

S Ananthakumaran[2]
Associate Professor, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation Vaddeswaram - 522502, Guntur Andhra Pradesh, India

***Abstract*—Over the most recent couple of years, the greater part of the information, for example books, recordings, pictures, clinical, forensic, criminal and even the hereditary data of people are being pushed toward digitals and cyber-dataspaces. This problem requires sophisticated techniques to deal with the vast amounts of data. We propose a novel solution to the problem of gaining actionable intelligence from the voluminous existing and potential digital forensic data. We have formulated an Automated Learning Framework ontology for Digital Forensic Applications relating to collaborative crime analysis and prediction. The minimum viable ontology we formulated by studying the existing literature and applications of Machine learning has been used to devise an Automated Machine Learning implementation to be quantitatively and qualitatively studied in its capabilities to aid intelligence practices of Digital Forensic Investigation agencies in representing, reasoning and forming actionable insights from the vast and varied collected real world data. A testing implementation of the framework is made to assess performance of our proposed generalized Smart Forensic Framework for Digital Forensics applications by comparison with existing solutions on quantitative and qualitative metrics and assessments. We will use the insights and performance metrics derived from our research to motivate forensic intelligence agencies to exploit the features and capabilities provided by AutoML Smart Forensic Framework applications.**

Keywords*—Forensic investigation; digital forensic; automated machine learning; smart forensic framework

I. Introduction

The overall utilization of portable savvy gadgets has expanded dramatically in the course of recent decades and now is essential in the running and preservation of every aspect of our day to day life. The gadgets range from the assortment of devices that incorporate cell phones, tablets, GPS, etc. The ubiquity of these digital gadgets is expanded essentially because of their utility, immense capacities and abilities and furthermore the depreciation in their prices as production becomes cheaper. Subsequently, they can hold the huge measure of business and private client's information. These gadgets are now a fundamental aspect of our day by day life since they contain private and basic data of clients [1]. In any case, these gadgets are additionally helpless against aggressors and are regularly turning into the significant vector of crimes, IP burglary, interruptions, security dangers, counterfeit reproductions and identity theft and etc. The quantity of advanced wrongdoings similarly increments as the new innovations, for example advanced gadgets and web increments. Therefore, these devices are turning into the vulnerable objectives for different sorts of cybercrimes and advanced assaults. However, Thanks to advancements in the field of Digital Forensics we get ways to also combat the ever encroaching aspect of criminal wrongdoing that pervades into our lives.

We will use a working definition for the term Digital Forensics Investigations (DFI) as, "The use of objective analysis toward the conservation, aggregation, validation, recognition, interpretation, documentation and representation of digital evidence got from digital sources for the intent of reconstruction of occurrences found to be illegal, or helping to predict events shown to be disruptive to peace or functioning of society".

II. Research Process Outline

Our study will be conducted in three phases:

1) Outlining a divergent meta study of the current research in the field ranging in their application potential and efficacy. This is outlined in our survey and commentary on the mentioned relevant literature on the subject. This will inform us in our second phase in finding applications and testable implementations of principles outlined. We will also use this to construct an accessible minimum viable ontology for collaborative and universal DFI practices based on the insights of the survey.

2) The assessments we made in the first phase will be used to propose architecture for our machine learning experimentations in this phase. We will describe the algorithms and review their capabilities amongst themselves. We will expand upon our methodology of collecting records of forensic & criminal data to test out the algorithms in terms of the accuracy in dealing with these large datasets.

3) From the results obtained by the techniques outlined in phase II, we conduct further review and analyses on the practicality of application and efficacy of the algorithms and also give further commentary on the contextual advantages and disadvantages of the analysed techniques. This will lead into using the learnings from the previous phases of research to implement a version of our proposed framework. We will

then use this implementation to comment on real world use case scenarios and test the practical feasibility of such a formulation of Forensic Framework after validation of proposed system along the mentioned quantitative and qualitative performance indices.

We will then finally conclude consolidating our learnings and insights during the process of our research and implementation and outline the scope and certain key directions for future research for our problem definition.

III. Literature Survey

Literature survey is a valuable step in software development workflows. Here we outline the general conceptions of machine learning approaches to this problem of parsing varied and voluminous digital and criminal forensic data for knowledge representation and getting actionable insights for DFI practices [2].

Knowledge Discovery shows smart computing at its finest, and is an interesting end-product of advancements in Information Technology. The ability to parse and to extract intelligence from data is a task that is of crucial importance to many fields of human development [3]. There is a lot of hidden synthesis waiting to be uncovered, this is the potential created by today's surplus of rich data. Data Mining and Knowledge Discovery Handbook, Second Edition organizes some current ideas, theories, notions, methodologies, and applications of data mining and knowledge discovery in databases (KDD) into a unified and comprehensive repository. Such KDDs provide additional intelligence utility ranging from preliminary on-field forensic assessments to mobile network flow and community cluster analysis.

Tensor Flow is an interface for expressing algorithms in machine learning and an implementation for such algorithms to be implemented. On a wide range of heterogeneous systems, from mobile devices such as phones and tablets to large-scale distributed systems with hundreds of machines and thousands of computing devices such as GPU cards, computations represented using machine learning can be performed with little to no modification, This paper describes the machine learning interface and an implementation of that interface that they have built at Google [4]. The system is versatile and may be accustomed specific a good sort of algorithms, as well as coaching and logical thinking algorithms for deep neural network models, and it's been used for conducting analysis and for deploying machine learning systems into production across over a dozen areas of engineering and alternative fields, as well as mechatronics, speech recognition, data retrieval, computer vision, natural language processing (NLP), geographic data extraction, and automated drug discovery.

Examining Deep Learning Architectures for Crime Classification and Prediction, A detailed study is presented on the classification and prediction of crime utilising deep learning architectures [5]. We analyse the efficacy of deep learning algorithms in this field and include suggestions for the design and training of deep learning systems using open data from police reports to predict areas of crime. A comparative analysis of 10 state-of-the-art methods against 3 different deep learning configurations is performed as a training data time series of crime types per venue. We show that the deep learning-based methods consistently outperform the current best-performing methods in our experiments with five publicly accessible datasets. In addition, in the deep learning architectures, we evaluate the effectiveness of different parameters and provide insights for configuring them in order to achieve improved performance in the classification of crime and ultimately prediction of crime.

H2O is machine learning and data analysis applications. A number of well-known businesses are using H2O for their processing of big data, and over 5000 organizations are currently using it, the website states. The main things H2O brings to R and Python developers, who already feel they have all the machine learning libraries they need, are ease of use and efficient scalability for datasets that are too large to fit into a large machine's memory. One of the things that make the H2O APIs so efficient and simple to use is that a large part of their interface is common to each of the machine learning algorithms. This allows the ensemble to model different learning architectures via trained submodels [6]. H2O also offers some changes in the quality of life, such as being able to manually or parametrically stop training until the model achieves acceptable user quality. It also comes with scalable and robust algorithm implementations, such as a multitude of feature encoding options, modelling options, hyperparameter tuning, scoring, etc. These algorithms also help to exploit the advantages of cluster computing and model ensemble preparation, but H2O lacks complete support for options for GPU computing. Although the latter can be applied via Java or C++ or via the implementations of TensorFlow.

Review: From our meta study on these publications and their methodologies and proposed architectures, we will be taking some of these approaches to DFI and ensure that our proposed system will be able to provide similar capabilities. Before building the proposed system, the techniques and consideration from these papers are also taken into account for the experiment implementation and validation.

IV. Proposed Work

The assessments we made in the Literature Survey will be used to propose architecture for our machine learning experimentations in this phase. We will describe the ontology modelling and the pipeline methodology and review their capabilities. We will expand upon our methodology of collecting records of forensic & criminal data to test out the algorithms.

A. Operational Ontology

In each step of the process, the system is based on Machine Learning principles and uses AI to ensure that decisions made by instruments have minimal false positives. However, a 100% precise and intelligent system cannot be conceptualised, the possibilities cannot be ignored for errors and false positives. Rigorous testing before use will be needed for the working model. While user inputs can be minimised at all possible levels, it is desirable to verify at each step to avoid errors, especially the system-generated report should be validated and cross-checked with objects before it is submitted to the court of law.

An abstract representation of our proposed Smart Machine Learning Digital Forensic framework is shown in Fig. 1.

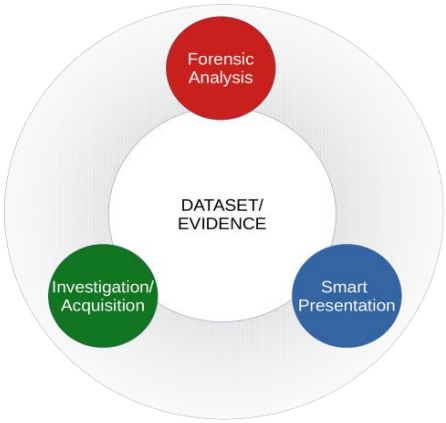

Fig. 1. Smart Digital Forensic Framework.

The proposed system is case-based and is considered to be a single package capable of resolving all three digital forensic method steps [7]. Most of the current instruments support these three measures, but they lack the rich interoperable intelligence described in our proposed system formulation, and this is the drawback we want to mitigate. Our structure is built with widely recognized traditional programming, AI and ML processes and toolkits, where existing data sets from previous forensic investigations are trained in the framework. These sets of data are useful for the system to understand what decision to take in which case. The probability of integration of AI at each point is discussed in the following subsections. In this context, the measures are called smart because they act on the basis of their knowledge and learning from it. Each phase needs to be retrained after training to see reliable outcomes. The test data sets can be used to validate the learning process and the instrument can be rigorously trained with more training data sets based on contextual performance metrics needed [11].

B. Automated Machine Learning (AutoML)

The most important aspects of making a prediction model is being able to use domain expertise and iterated learning (either by an intelligent human agent or meta learning methods) to find out what the important features are for the prediction task and how to optimize the hyperparameters of the learning algorithms for successful learning over large databases, which cannot be pruned or cleaned by a human data scientist.

We will be using the H2O AutoML framework for our proposed implementation. It is capable of doing Categorical Ensembling in a live production environment. This allows it to effectively combine multiple trained pipelines and use the combined data of previous user runs of Smart Forensic (SF) Analysis as shown in Fig. 2, where new data is appended to trained models of previous runs.

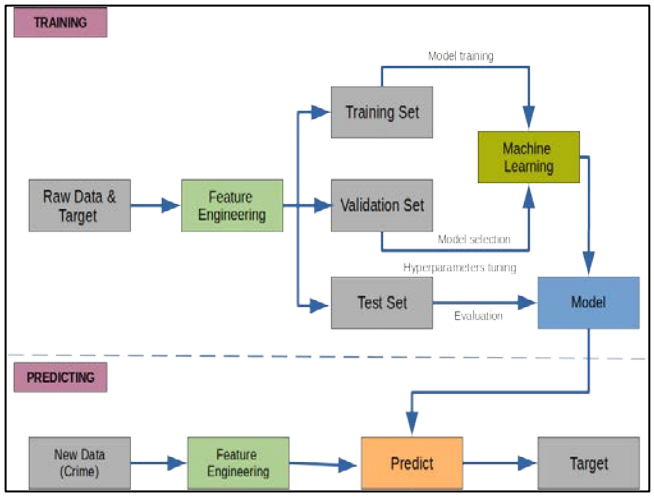

Fig. 2. Experiment Pipeline Steps.

This is where good implementations of AutoML pipelines will help us in satisfying the basic requirements for our SF framework. In addition, open-source libraries that implement AutoML techniques are available, focusing on the particular data transformations, models, and hyperparameters used in the search space and the types of algorithms used to traverse or optimise the possibilities of the search space, with the most popular being variants of Bayesian Optimization [8]. However by using H2O we also get access to state-of-the-art techniques like pruned decision trees, Gradient Boosting and even tuned Deep Learning Algorithms and advanced encoding and feature preprocessing methods.

There are hyperparameters in any machine learning system, and the most basic task in AutoML is to set these hyper-parameters to optimise performance automatically.

It has capabilities to:-

- Reduce the human effort needed for machine learning to be implemented. In the sense of AutoML, this is especially important.
- Improve the efficiency of machine learning algorithms (by adapting them to the problem at hand); this has led to new state-of-the-art performances in many studies for significant machine learning benchmarks.
- Improving the reproducibility of scientific experiments and their justice. Clearly, automated HPO is more reproducible than manual search. It makes reasonable comparisons simpler since different approaches can only be equally compared if they all obtain the same degree of tuning at hand for the problem.

Fig. 3 shows the description of processes through the lens of our SF Acquisition pipeline.

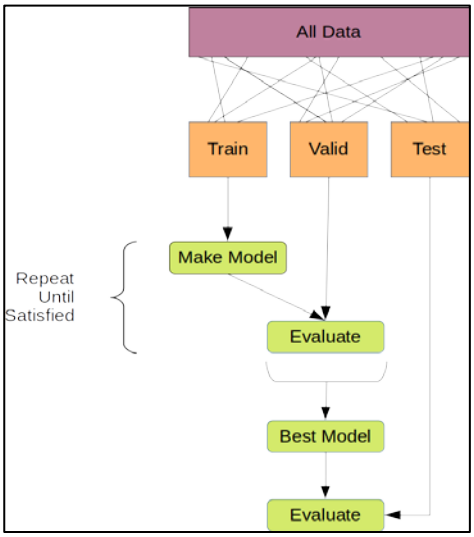

Fig. 3. SF Acquisition Pipeline.

These algorithms are also robust for transfer learning implementations and combining dataframes for better modelling and richer feature space definition derived from the merged datasets. Fig. 4 shows the overview of the server architecture used for our work.

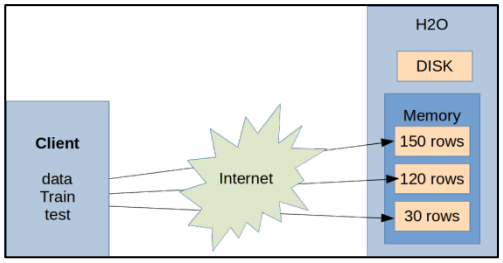

Fig. 4. Client Server Architecture.

C. Implementation Specification

We posit that an H2O based implementation would be extremely effective for SF Analysis and SF Acquisition and by an effective user interface as per the UX pertaining to different use case and ethno-social differences. A SF application based on our ontology and framework description seems promising for the tasks of distributed data mining and forensic analysis for agencies and Investigator across all levels of hierarchy upto policy makers and field agents. As the autoML computation pipeline also supports TensorFlow which also enables distributed computing applications [9]. However, this is not supported "out of the box" and requires additional engineering to implement robustly.

H2O's REST API allows all H2O capabilities to be accessed via JSON over HTTP from an external programme or script. H2O's web interface (Flow UI), R binding (H2O-R), and Python binding are used for the rest of the API (H2O-Python).

We will use its REST API via its driverless.ai cloud implementation to construct our pipeline via the "codeless" interface provided by the proprietary driverless.ai web interface through a 2 hour evaluatory trial that can be accessed here https://www.h2o.ai/try-driverless-ai/.

D. Experiment Overview

We built a LightGBM Model using Driverless AI to predict RAPE given 32 original features from the input dataset "01_District_wise_crimes_committed_IPC_2001_2012.csv". This regression experiment was completed in 41 minutes and 13 seconds (0:41:13), using 1 of the 32 original features, and 2 of the 2 engineered features.

E. Data Overview

The dataset is obtained from www.data.gov.in, a government website; the data being provided by the National Crime Records Bureau (NCRB).

9017 rows and 13 columns of 1.3 MiB file size are included in the crime datasets we used for our experiment.

F. Experiment Pipeline

For this experiment, Driverless AI performed the following steps (shown in Fig. 5) to find the optimal final model:

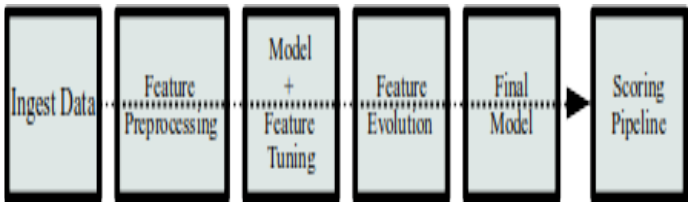

Fig. 5. Experiment Pipeline Steps.

The steps in this pipeline are described in more detail below:

1) Ingest data: detected column types

2) Feature preprocessing: turned raw features into numeric

3) Model and feature tuning: This stage combines random tuning of hyperparameters with selection and generation of characteristics. Features are modified in each iteration using variable significance as a probabilistic from the previous iteration before determining what new features to build. The best performing model and features are then passed to the feature evolution stage.

- Identified the optimal parameters for constant, decision tree, lightgbm and xgboost methods by training models with different variables.
- The best parameters will be those who produce the least root mean square error (RMSE) on the internal validation data.
- To evaluate features and prediction models, 105 models were trained and scored.

4) Feature evolution: To find the best set of model parameters and feature transformations to be used in the final model, this stage uses a genetic algorithm.

- Found the best representation of the data for the final model training by creating and evaluating 2 features over 34 iterations.

- Trained and Scored 116 models to further evaluate engineered features.

5) Final model: created the best model from the feature engineering iterations.

- No stacked ensemble is done because a time column was provided.

6) Create scoring pipeline: created and exported the Python scoring pipeline

Driverless AI trained models throughout the experiment in an effort to determine the best parameters, model dataset, and optimal final model. The processes are shown in Table I.

TABLE I. DRIVERLESS AI EXPERIMENT STAGES

Driverless AI Stage	Timing (seconds)	Number of Models
Data Preparation	20.81	0
Model and Feature Tuning	799.57	105
Feature Evolution	1,322.19	116
Final Pipeline Training	142.09	1

G. Experiment Settings

Table II shows the settings selected for our experiment. The Defined Parameters represents the high-level parameters.

TABLE II. PARAMETERS AND SELECTED VALUES

Parameter	Value
num prediction periods	1
num gap periods	0
accuracy	7
time	5
gpus enable	True
is image	False
seed	False
is timeseries	True
is classification	False
interpretability	interpretability

H. Supported Algorithms

We specify here the ML algorithms used as candidate models for our AutoML SF Framework which includes the following algorithms.

1) LightGBM: LightGBM is a Microsoft-developed gradient boosting framework that uses tree-based learning algorithms. It was specifically designed for lower memory consumption and greater performance and faster training speed. Analogous to XGBoost, it is among the highest quality implementations for gradient boosting. It also is used within Driverless AI for the fitting of Random Forest, DART (experimental algorithm), and Decision Tree approaches.

2) XGBoost: XGBoost is a supervised learning model that enacts a method to make accurate models called boosting. Boosting alludes to the ensemble teaching technique of sequentially constructing many models, with each latest model attempting to rectify the inadequacies in the previous version [10]. In tree boosting, a decision tree is each new model that is added to the ensemble. XGBoost offers parallel tree boosting (identified as GBDT, GBM) that quickly and accurately accomplishes many challenges of data science. XGBoost is one of today's best gradient boosting machine (GBM) frameworks for several issues. Driverless AI implements the DART (experimental algorithm) methods of XGBoost GBM and XGBoost.

3) Decision Tree (DT): A DT is a binary (single) tree model dividing the population of training data into leaf nodes (sub-groups) with consistent conclusions. No column or row sampling is undertaken, and hyper-parameters [12] monitor the depth of the tree and the growth method (depth-wise or loss-guided).

4) Constant model: The algorithm Constant Model predicts same constant value for any input data. By optimising the given scorer, the constant value is computed. For example, for MSE/RMSE, the constant is the target column's (weighted) mean. It is the (weighted) median for MAE. For other scorers, such as MAPE or custom scorers, an optimization process finds the constant. For classification issues, the constant probabilities are the priors identified. A constant model is considered as a baseline reference model. A warning will be issued if it ends up being used in the final pipeline, because it shows vulnerability in the dataset or target (e.g. in attempting to predict a stochastic possibility).

5) Follow The Regularized Leader (FTRL): A DataTable implementation [13] of the FTRL-Proximal online learning algorithm suggested in [16] is Follow the Regularized Leader (FTRL). For parallelization, such an implementation utilises a hashing trick and a Hogwild approach [15]. For categorical targets, FTRL facilitates binomial and multinomial classification, along with regression for continuous targets.

6) RuleFit: Through first adapting a tree model and then fitting a Lasso (L1-regularized) GLM model, the RuleFit [14] algorithm creates an optimum set of decision rules to create a linear model composed of the most crucial tree leaves (rules).

V. RESULTS

We will now display the quantitative metrics of our implemetation’s performance on the prediction and analysis tasks. Further in the discussion we will elucidate briefly on the performance of the implementation on its quantitative metrics and further comment on its capabilities and viability with respect to qualitative concerns of DFI and how our formulation fares in those regards.

A. Model Tuning

Driverless AI automatically split the data into training and validation data, ordering the data by YEAR. The experiment predicted 131536000 seconds ahead with no gap between training and forecasting.

Table III shows the score and training time of the constant, decision tree, lightgbm and xgboost models evaluated by AI. Following table also shows the top 10 parameter tuning models evaluated, ordered based on a combination of least score and lowest training time.

TABLE III. SCORES AND TRAINING TIME OF ALGORITHMS

job order	booster	nfeatures	scores	training times
17	lightgbm	127	10.021	11.7719
10	lightgbm	39	10.0522	10.5283
1	lightgbm	38	38.2283	12.4251
3	lightgbm	33	39.3688	7.5582
19	lightgbm	116	45.1921	11.2392
21	lightgbm	98	51.531	8.3811
15	gbtree	66	58.8309	15.0382
13	lightgbm	70	60.9843	7.9107
4	gbtree	33	82.8789	6.5336
16	decision tree	77	90.2014	5.2849

More detailed information on the parameters evaluated for each algorithm is shown in the following tables, (Table IV Constant tuning, Table V Decision Tree tuning, Table VI LightGBM tuning and Table VII gbtree tuning).

B. Feature Evolution

During the Model and Feature Tuning Stage, we evaluate the effects of different types of algorithms, algorithm parameters, and features. The goal of the Model and Feature Tuning Stage is to determine the best algorithm and parameters to use during the Feature Evolution Stage.

In the Feature Evolution Stage, Driverless AI trained lightgbm models (116) where each model evaluated a different set of features. The Feature Evolution Stage uses a genetic algorithm to search the large feature engineering space. The graph in Fig. 6 shows the effect the Model and Feature Tuning Stage and Feature Evolution Stage had on the performance.

TABLE IV. CONSTANT TUNING

job order	booster	nfeatures	scores	training times
23	constant	1	215.4763	2.1245

TABLE V. DECISION TREE TUNING

tree method	grow policy	max depth	max leaves	nfeatures	scores	training times
gpu_ hist	depth wise	8.0	128.0	77	90.202	5.285
gpu_ hist	loss guide	6.0	128.0	58	90.205	4.977
gpu_ hist	loss guide	8.0	64.0	74	90.208	5.282
gpu_ hist	loss guide	10.0	128.0	91	90.268	5.299
gpu_ hist	loss guide	4.0	32.0	35	90.319	4.706

TABLE VI. LIGHTGBM TUNING

tree method	grow policy	max depth	max leaves	n features	scores	training times
gpu_ hist	depthwise	6.0	0.0	127	10.021	11.772
gpu_ hist	depthwise	6.0	0.0	39	10.053	10.529
gpu_ hist	lossguide	0.0	1024.0	38	38.228	12.426
gpu_ hist	depthwise	10.0	0.0	33	39.369	7.5582
gpu_ hist	depthwise	10.0	0.0	116	45.192	11.239
gpu_ hist	loss guide	0.0	1024.0	98	51.531	8.3811
gpu_ hist	loss guide	0.0	1024.0	70	60.985	7.9107
gpu_ hist	depthwise	6.0	0.0	35	10.587	10.116
gpu_ hist	depthwise	10.0	0.0	38	10.645	12.5988
gpu_ hist	depthwise	6.0	0.0	35	11.566	9.3648

TABLE VII. GBTREE TUNING

tree method	grow policy	max depth	max leaves	nfeatures	scores	training times
gpu_ hist	loss guide	0.0	1024.0	66	58.83	15.04
gpu_ hist	depth wise	10.0	0.0	33	82.88	6.534
gpu_ hist	loss guide	0.0	1024.0	111	112.49	5.176
gpu_ hist	depth wise	6.0	0.0	35	26.821	23.9731

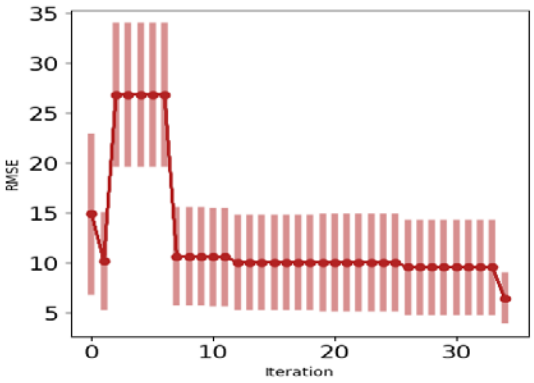

Fig. 6. Feature Evolution Graph.

C. Feature Transformations

Table VIII, ordered by value, the top features used in the final model are shown. The characteristics in the table are limited to the top 50 and are restricted to those with relative significance equal to or greater to 0.003. The function is an original column if no transformer has been applied.

TABLE VIII. TOP FEATURES USED IN FINAL MODEL

no.	Feature	Description	Transformer	Relative Importance
1	29_InteractionAdd: CUSTODIAL RAPE: OTHER RAPE	[CUSTODIAL RAPE] + [OTHER RAPE]	Interaction	1.0
2	22_OTHER RAPE	OTHER RAPE (Original)	None	0.9322
3	29_InteractionSub: CUSTODIAL RAPE: OTHER RAPE	[CUSTODIAL RAPE] - [OTHER RAPE]	Interaction	0.5503

Fig. 7 shows the bar graph of Features and Relative Feature Importance.

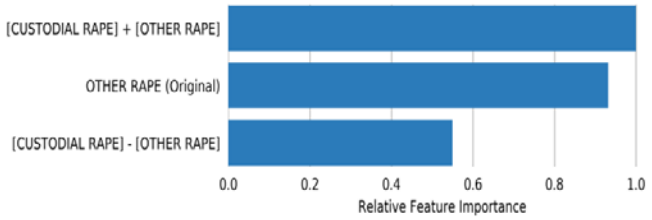

Fig. 7. Features and Relative Importance.

D. Final Model

Final pipeline of LightGBMModel with ensemble level is equal to 0 Transforming 30 initial characteristics. In each of 1 model, 3 characteristics each suit on time-based hold-out. Fig. 8 shows the Final Model Pipeline Feature.

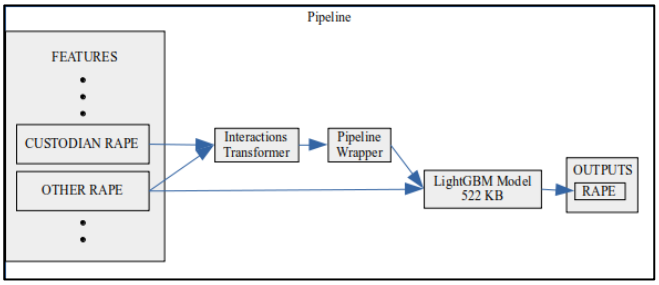

Fig. 8. Feature to Pipeline.

Details:

- The fitted features of the final model are the best features found during the feature engineering iterations.
- The target transformer indicates the type of transformation applied to the target column.

Table IX describes final model transformation details. Model Index: 0 has a weight of 1 in the final ensemble.

TABLE IX. FINAL MODEL TRANSFORMATION DETAILS

Model Index	Type	Model Weight	Fitted features	Target Transformer
0	LightGBMModel	1	3	log

TABLE X. PERFORMANCE OF FINAL MODEL

Scorer	Better score is	Final ensemble scores on validation (internal or external holdout(s)) data	Final ensemble standard deviation on validation (internal or external holdout(s)) data
RMSE	lower	6.478455	2.335938

Performance of the final model is shown in Table X. The scorer we used here is RMSE.

Fig. 9 shows the graph of performance for Actual vs predicted.

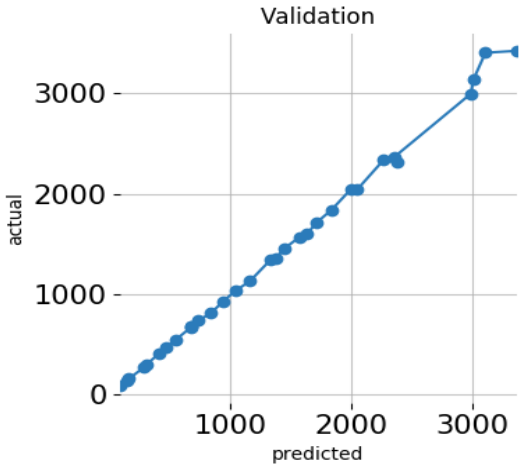

Fig. 9. Performance (Actual vs Predicted).

E. Alternative Models

During the experiment, we trained 27 alternative models using Driverless AI. The following Table XI shows the algorithms were evaluated during our experiment.

An array of algorithms, including but not limited to the Constant, Decision Tree, Light GBM, gbtree, XGBoost GLM, XGBoost GBM, XGBoost Dart, RuleFit, Tensorflow, and FTRL models, can be tested by Driverless AI. Table XII below illustrates why, if any, such algorithms for the final model were not chosen.

TABLE XI. DETAILS ABOUT ALGORITHMS EVALUATED IN EXPERIMENT

algorithm	package	version	documentation
constant	custom package	1.9.0	reference model that predicts a constant aimed at minimizing the given scorer
decision tree	light gbm	2.2.4	LightGBM, Light Gradient Boosting Machine. Contributors: https://github.com/microsoft/LightGBM/graphs/contributors.
lightgbm	light gbm	2.2.4	LightGBM, Light Gradient Boosting Machine.Contributors: https://github.com/microsoft/LightGBM/graphs/contributors.
gbtree	xgboost	1.1.0	XGBoost: eXtreme Gradient Boosting library.Contributors: https://github.com/dmlc/xgboost/blob/master/CONTRIBUTORS.md

TABLE XII. DETAILS ABOUT ALGORITHM SELECTION IN FINAL MODEL

algorithm	selection
gblinear	algorithm not evaluated due to experiment configuration
rulefit	algorithm not evaluated due to experiment configuration
tensorflow	algorithm not evaluated due to experiment configuration
ftrl	algorithm not evaluated due to experiment configuration
dart	algorithm not evaluated due to experiment configuration
gbtree	Due to low performance during the model tuning process, not selected
decision tree	Due to low performance during the model tuning process, not selected
lightgbm	selected for final model

F. Deployment

For our experiment, Python Scoring Pipelines are available for productionizing the final model pipeline for a given row of data or table of data.

Python Scoring Pipeline pack provides an assembled model and samples of the Python 3.6 code base to generate models designed with H2O Driverless AI. Below is the Python Scoring Pipeline:

- admin/h2oai_experiment_21c8e18c-059c-11eb-833e-0242ac110002/scoring_pipeline/scorer.zip.

In this package, the files allow us to transform and score new data in a few different ways:

- We can import a scoring module from Python 3.6, then use the module to convert and score on updated data.
- We can use the TCP/HTTP scoring service included with this package for other applications and scripts to call the scoring pipeline module via remote procedure calls (RPC).

VI. DISCUSSION

We have provided the quantitative measurements as a comparison between the algorithms as they offer multiple varied expressions of our SF Framework and the real world value of the three implementations but they cannot be assessed by raw performance alone [17]. However, the general improvements provided by an AutoML engine can be assessed from the quantitative results provided. There is significant improvement in training time and prediction accuracy because of the appropriate algorithm selection and hyper-parameter optimization capabilities of the AutoML engine. This also validates our assumption that H2O autoML can be a well-rounded candidate for a simple and generalized metalearner for DFI by using our SFI Framework Ontology.

The report generated by SF framework also provided the performance metrics we have referenced in this paper. This generated report demonstrates the SF Report Generation aspect of a well-rounded SF Framework. One of the main scaling problems of our traditional legal is justice and bureaucratic sluggishness. The major factor of this bottleneck is the inability of human agents involved in such institutions to process the data effectively and manually create paperwork. If we augment the SF Reporting with scripts and templates to interface with existing legal protocols we will be able to greatly improve efficiency and efficacy of forensic agencies, with minimal feature re-tooling. Such elegant yet exhaustive SF Acquisition and SF Reporting implementation provides an easy way to bridge the gap and aid the traditional institutions to translate and transition into more appropriate mechanisms and institutions for our current needs for law enforcement and judicial systems.

Hence, we see how well our H2O implementation fares in our proof of concept in a well-rounded qualitative assessment of its capabilities in our three pronged ontology of SF Framework. We have provided the quantitative measurements as a comparison between the algorithms as they offer multiple varied expressions of our SF Framework and the real world value of the three implementations but they cannot be assessed by raw performance alone. We posit that H2O autoML provides us the most well rounded candidate for a simple and generalized metalearning for DFI by using our SFI Framework Ontology. Such smart report generation capabilities of the DFI framework is crucial in avoiding opaque and dangerous black boxes and can help illustrate the workings and reasoning behind the models learning biases. This is extremely important for real world applications in judicial or criminal and forensic use.

VII. CONCLUSION

The topic of DFI is increasingly complex, and is blossoming to be a field that usually requires a huge abundance of complex data to be parsed and acquired from the scene of forensic interest. DFI practices include evaluating the digital evidence about the committed crime to be used as legal proof in the court of law. In this cycle, AI can be seen as an ideal way to deal with and take care of the issues that exist in the computerized criminology field. Different AI calculations and strategies can be valuable during the time spent separating and breaking down computerized proof. Automates Machine Learning Frameworks will improve this cycle by managing a lot of information in a brief timeframe range. It is clear that these improvements will be capable of providing solutions for taking vast volumes of forensic data and representing the data to make practical and highly intelligent investigative actions and prosecution decisions with a high degree of precision and good outcome consistency. In the forensic analysis phases, investigators are encouraged to use these methods as they give them the opportunity to counter various forms of crimes far beyond what is actually capable of doing so.. Well defined and minimum viable collaborative ontologies of Digital Forensic Investigations, provide avenues for advancements in the field of Machine Learning and Artificial Intelligence a way to be incorporated with ease into traditional Criminal and Forensic Agencies.

REFERENCES

[1] A. Guarino, "Digital forensics as a big data challenge," in ISSE 2013 securing electronic business processes: Springer, 2013, pp. 197-203.

[2] Iqbal, Salman & Alharbi, Soltan. (2019). Advancing Automation in Digital Forensic Investigations Using Machine Learning Forensic 10.5772/intechopen.90233. I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[3] Maimon, Oded & Rokach, Lior. (2010). Data Mining and Knowledge Discovery Handbook, 2nd edition.

[4] TensorFlow (2016): Large-Scale Machine Learning on Heterogeneous Distributed Systems.

[5] Stalidis, Panagiotis & Semertzidis, Theodoros & Daras, Petros. (2018). Examining Deep Learning Architectures for Crime Classification and Prediction.

[6] Practical Machine Learning with H2O by Darrencook, Released December 2016, Publisher(s): O'Reilly Media, Inc. ISBN: 9781491964606.

[7] Rughani, Dr. Parag. (2017). Artificial Intelligence Based Digital Forensics Framework. International Journal of Advanced Research in Computer Science. 8. 10-14.10.26483/ijarcs.v8i8.4571.

[8] Z. Li et al., "A Blockchain and AutoML Approach for Open and Automated Customer Service," in IEEE Transactions on Industrial Informatics, vol. 15, no. 6, pp. 3642-3651, June 2019, doi:10.1109/TII.2019.2900987.

[9] J. P. Ono, S. Castelo, R. Lopez, E. Bertini, J. Freire and C. Silva, "PipelineProfiler: A Visual Analytics Tool for the Exploration of AutoML Pipelines," in IEEE Transactions on Visualization and Computer Graphics, vol. 27, no. 2, pp. 390-400, Feb. 2021, doi: 10.1109/TVCG.2020.3030361.

[10] Z. Wen, J. Shi, B. He, J. Chen, K. Ramamohanarao and Q. Li, "Exploiting GPUs for Efficient Gradient Boosting Decision Tree Training," in IEEE Transactions on Parallel and Distributed Systems, vol.30, no. 12, pp. 2706-2717, 1 Dec. 2019, doi: 10.1109/TPDS.2019.2920131.

[11] Yuki, Jesia & Sakib, Md. Mahfil & Zamal, Zaisha & Habibullah, Khan & Das, Amit. (2019). Predicting Crime Using Time and Location Data. 124-128. 10.1145/3348445.3348483.

[12] J. R. J. M. l. Quinlan, "Induction of decision trees," vol. 1, no. 1, pp. 81-106, 1986.

[13] DataTable for Python, Z. https:github.com/h2oai/datatable.

[14] J. Friedman, B. Popescu. “Predictive Learning via Rule Ensembles”. 2005. http://statweb.stanford.edu/~jhf/ftp/RuleFit.pdf.

[15] Niu, Feng, et al. “Hogwild: A lock-free approach to parallelizing stochastic gradient descent.” Advances in neural information processing systems.2011.https://people.eecs.berkeley.edu/~brecht/papers/hogwildTR.pdf

[16] McMahan, H. Brendan, et al. “Ad click prediction: a view from the trenches.” Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2013. https://research.google.com/pubs/archive/41159.pdf.

[17] Agarwal R., Kothari S. (2015) Review of Digital Forensic Investigation Frameworks. In: Kim K. (eds) Information Science and Applications. Lecture Notes in Electrical Engineering, vol 339. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-46578-3_66.