Secure Routine

A Routine-Based Algorithm for Drivers Identification

Davide Micale*, Gianpiero Costantino†, Ilaria Matteucci†, Giuseppe Patanè‡ and Giampaolo Bella*
*University of Catania, Dip. di Matematica e Informatica, Catania, Italy
†Consiglio Nazionale delle Ricerche (CNR), Istituto di Informatica e Telematica (IIT), Pisa, Italy
‡Park Smart Srl, Catania, Italy
Email: davide.micale@phd.unict.it, name.surname@iit.cnr.it, giuseppe.patane@parksmart.it, giamp@dmi.unict.it

***Abstract*—The introduction of Information and Communication Technology (ICT) in transportation systems leads to several advantages (efficiency of transport, mobility, traffic management). However, it may bring some drawbacks in terms of increasing security challenges, also related to human behaviour. As an example, in the last decades attempts to characterize drivers' behaviour have been mostly targeted. This paper presents *Secure Routine*, a paradigm that uses driver's habits to driver identification and, in particular, to distinguish the vehicle's owner from other drivers. We evaluate Secure Routine in combination with other three existing research works based on machine learning techniques. Results are measured using well-known metrics and show that Secure Routine outperforms the compared works.**

Keywords–driver identification; secure routine; machine learning; automotive.

I. Introduction

Modern vehicles can be considered as computer on wheels. The mechanical parts are often controlled by software components and communication protocols are in charge of exchanging data among vehicle's components. For this reason, modern vehicles are Cyber Physical Systems *(CPS)* in which used technologies bring countless advantages in terms of, for instance, efficiency of city operations and services. An example among all is the Internet connectivity. Within this context, a problem of particular interest is how to leverage vehicular and/or smartphone data to characterize driver identification. Its characterization finds application in the development of software, which can be used by insurance companies to check and identify drivers or, for instance, to discourage auto theft. In 2019, around 56k vehicles were targeted by thieves in UK [1]. It equates to one car stolen every 9 minutes and 45% of thefts occurred between midnight and 6 AM. Having a strategy to classify the driver's behaviour may help to mitigate this trend.

Routine based classification is a type of classification [2] that aims to find actions that are frequently repeated in time. To complete a task, people repeat sequence of actions previously saw from others or done by themselves, no matter how tough the task is [3]. Two persons may accomplish the same task with similar actions but with little fundamental differences [3]. Routines can describe how people organize their lives: daily commute, weekly, meetings, holidays. Routines can also describe how a driver approaches to an intersection [4].

Based on these aspects of routine, here we introduce the paradigm of Secure Routine *(SR)* that takes into account not only what the user does but also how much frequently. We use the SR paradigm within the automotive context with the aim to classify drivers. To achieve this, we elaborate and implement the SR algorithm that exploits sensors' car data, obtained, for instance, through the *OBD-II* [5] diagnostic port. The SR algorithm evaluates the recorded data and, in particular, uses the timestamp to make an accurate classification of drivers. Then, SR leverage a Machine Learning (*ML*) technique to establish driver's routines and to properly identify the driver.

To test the goodness of the Secure Routine algorithm, we compare it with other research works present in literature. The comparison is done on two different datasets and the results are evaluated using three metrics: *Accuracy*, *Precision* and *Recall*. Findings show that Secure Routine outperforms the compared works in all the tests carried out.

The paper is structured as follows: next section presents the state of the art. In Section III, we introduce the background on ML techniques. In Section IV, we present the Secure Routine paradigm used to identify drivers. Then, in Section V, we compare Secure Routine with other research works presented in literature. Finally, Section VI draws the conclusion of this paper and presents some hits for future research directions.

II. State of the Art

In literature, there are several solutions based on ML techniques for the identification of driver's behaviour. Bernardi et al. [6] used a Multi Layer Perception (MLP) to identify drivers. They used three datasets obtaining respectively 94%, 95% and 92% of Accuracy. In particular, these results were obtained using a Start&Stop sliding window. A sliding window combines several consecutive instances in a single instance. In particular, Start&Stop joins instances starting when the car is moving until the car stops.

Gao et al. [7] discriminated drivers through Stop-and-Go events using a *voting strategy*. A Stop-and-Go event occurs when the car slowdowns until stops (stop phase), it stands still for five or more seconds and then speeds up (go phase).

Wang et al. [8] identified 30 drivers by using the voting strategy and Random Forest algorithm. Authors split data and tests into different window sizes. They use six sensor signals and three derived sensor's signals along with five statistical features. With 5 minutes of testing data this model achieves almost 93% of Accuracy. With a sliding window of 5 seconds and 6 minutes of testing data they achieve 100% of Accuracy.

Girma et al. in [9] used the Long Short-Term Memory (LSTM) algorithm with sliding windows and tested their model on [10] and [11] datasets with Precision and Recall of 98%.

Kwak et al. in [12] selected 15 features to identify drivers behaviour. For each feature they computed the mean, median and standard deviation according to a reference sliding win-

 ISBN: 978-1-61208-795-5

dow. Thus, the total number of features is 45. They used different ML algorithms and achieved the best Accuracy of 99,6% applying Random Forest on [10] dataset.

Martinelli et al. in [13] tested several Decision Tree algorithms with the same dataset [10] using all 51 features. They obtained a Precision and Recall equal to 99,2% with J48. The same authors in [14] used only six features out of 51 features of [10] dataset. In this case, Precision and Recall decreased to 98,9% due to under-fitting.

Compared to our paper, [6], [7], [8], [12], [13] and [14] do not look for frequency. Also, LSTM in [9] obtained lower scores in comparison with a *Decision Tree (DT)* algorithm ([12], [13] and [14]) on the same dataset. As shown by [14], certain features discriminate better than others for some drivers. Hence, SR must use the best feature set for each driver. [6], [14] and [13] are the only ones that make owner-driver identification, they select the same feature set for all drivers. Finally, SR breaks down the timestamp in fine grained units to detect frequency in order to increase the accuracy.

III. Machine Learning

ML is the study of computer algorithms that improve automatically through experience. ML algorithms build a mathematical model to make predictions or decisions without being explicitly programmed to do so. At the basis of the model, there is a dataset that has to be processed. Such dataset can be considered as a table in which all data are listed. Each row of the table is called *instance* and each column represents a *feature* of the instance. The dataset is usually split into two parts, the *training dataset* and the *test dataset*. The model is created on the basis of the training dataset. Instead, a *test dataset* represents all instances adopted to verify how much accurate our model is in doing the classification.

ML techniques are largely adopted for the identification and classification of users. In the following, we introduce an example of ML algorithm based on DT predictive modelling approach. A DT consists on a tree data structure that contains rules to classify the instance. For each level of the tree, the value of a feature of the instance is tested, for example, through a specific question. Each internal node of the tree contains a test. Depending on the answer, the model follows a different edge: the left edge if the result of the test is true, otherwise the right edge is followed. Finally, the leaf nodes, i.e., the nodes with no children, contain the prediction.

A. Decision Tree Requirements

A DT algorithm must create a tree with the minimum number of levels. This allows the ML algorithm to classify the instance as fast as possible. To build a DT with a low number of levels, it is necessary to select the best tests for the model. This is done by selecting the appropriate *Formula* to make the selection. A *Formula* specifies the criterion chosen to establish which is the next test to perform in the DT.

For instance, let *Alice* and *Bob* be two drivers that are used to going on the Sixth Avenue. Alice goes on the Sixth Avenue all days of the week, instead Bob goes only from Monday to Friday. Bob drives slightly faster than Alice, with a speed up to 55 Km/h. A possible DT model is the one in Figure 1(a) that is built by putting on the tree root the following test:

"*Is today Saturday or Sunday?*"

Following the root test, we have that the left child is taken by Alice instead the right child corresponds to the following test:

"*Is the vehicle speed lower than 55 Km/h?*"

Again the left child is a leaf node that represents Alice, whereas the right child is the leaf node representing Bob.

Despite the above DT model is a valid model for our example, we may produce a better tree in which a root node is configured with the following test (Figure 1(b)):

"*Is the vehicle speed lower than 55 Km/h?*"

In this case, the left child is the leaf node Alice and the right child is the leaf node Bob. Hence, a ML algorithm concludes its prediction with only one test.

A DT has to be simple. This allows the DT to be flexible enough to represent also further instances. Thus, if the built model is too complex, it may not represent new labelled instances, i.e., for instance those ones present in a test-set. This may cause a high error rates, generating the *over-fitting* error. To reduce the over-fitting error, the *pruning* technique can be adopted to obtain a simpler version of the tree by pruning some nodes. Another solution to mitigate the over-fitting error is the *feature selection* that works by removing features. However, pruning too many nodes and removing too many features or relevant ones may lead to higher error rates, aka *under-fitting*.

B. Decision Tree Algorithms

Several DT algorithms were developed to generate models. The *C4.5* was proposed in 1993 [15] and it uses the Gain Ratio (GR) of a feature "X" of the training set (T) to establish which is the best test to perform.

$$GR = \frac{H(T) - H(T|X)}{H(X)} \quad (1)$$

where:

- $H(T)$ indicates the *entropy* of T, i.e., the quantity of information carried by the probability distribution of labels in T [16], calculated as:

$$H(T) = -\sum_{j=1}^{k} \frac{freq(C_j, T)}{|T|} \times log_2 \left(\frac{freq(C_j, T)}{|T|} \right) \quad (2)$$

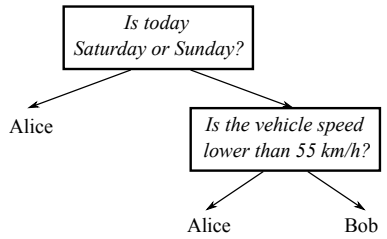
  where:
  - k is the number of classes;
  - $freq(C_j, T)$ is the number of instances in the j-th class;
  - $|T|$ is the number of instances of T.
- $H(T|X)$ indicates the entropy after partitioning T in "n" parts, where "n" is the number of possible values assumed by X:

$$H(T|X) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} \times H(T_i) \quad (3)$$

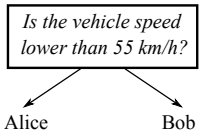
  where:
  - $|T_i|$ is the number of instances with the i-th value assumed by the feature X;
  - $H(T_i)$ indicates the entropy of the set of instances with the i-th value assumed by the feature X.
- $H(X)$ indicates the entropy of X:

$$H(X) = -\sum_{i=1}^{n} \frac{|T_i|}{|T|} \times log_2 \left(\frac{|T_i|}{|T|} \right) \quad (4)$$

 ISBN: 978-1-61208-795-5



(a) DT with two levels.



(b) DT with one level.

Figure 1. Comparison of two possible DT for solving the same problem.

Note that C4.5 can handle features with unknown values and real numbers and may make use of the pruning technique.

Random Forest (RF) [17][18] is an algorithm formed by a set of DTs. Each tree is built from a random sampling with replacement of the training-set. Each node of a tree is the best test defined on a subset of features, instead of on all available ones. Trees are not pruned. In prediction phase, an instance is run on each tree and each tree makes a prediction. The most predicted value becomes the prediction of RF. Also, RF includes a procedure in case of unknown values in the dataset.

IV. Secure Routine

In literature, the concept of *Routine* is already exploited to classify users or drivers [4]. A Routine is defined as a set of actions that a person frequently perform in response to a circumstance [19]. Hence, routines can describe how people organize their lives: daily commute, weekly, meetings, holidays. Here, we refine the concept of Routine by introducing the paradigm of *Secure Routine* that takes into account not only what the user does but also how much frequently.

We define SR and present its application into the automotive context to perform driver's behavioural identification. To this aim, SR analyses all tracking data recorded by vehicle's sensors while the user is driving it. Tracked data are organized in separate instances according to the sensor that collects them and the timestamps when the event occurs. Hence, SR firstly decomposes the timestamp of each instance and extracts second, minute, hour, day of week, day, month and year. Then, SR removes less relevant features, as we will describe below using the Feature Selection *(FS)* technique. Successively, the data collected by sensors are correlated with the timestamp previously decomposed. Then, a ML algorithm examines these data. The output is a model representing users' Secure Routine. As final step, the obtained model is compared with an observed user's driver behaviour for his/her identification.

To show the value added by the Secure Routine to identify drivers, we introduce the following example. Let us consider Alice and Bob who are used to going on the Sixth Avenue. Alice usually goes there at 12PM, and Bob at 7PM. If we do not consider the timestamp information, the resulting model of Alice and Bob will contain only the information *"The user is used to going on the Sixth Avenue"*. In this situation, the observed behaviour will be compared to understand whether the driver is Alice or Bob. However, this selection is quite difficult since the missing timestamp information is fundamental to distinguish between the drivers. On the contrary, if we consider also the timestamp in which the event happens, the identification will be unique in this case. In fact, if the vehicle is at 7PM on the Sixth Avenue, therefore the driver is Bob.

This is what Secure Routine does considering daily routines as well as monthly and yearly ones. Hence, SR may be very useful, for instance, to mitigate scenarios as the one depicted in Section I: in UK cars are often stolen at night. If the vehicle's owner does not usually drive during the night, SR can easily detect the weird behaviour. In particular, the SR paradigm is built upon a ML algorithm that uses as training-set the data recorded through an OBD-II device. A closer working mechanism of SR is presented in [20]. Here, the authors prefer to involve the interval between a rerun of the same action. Let us consider this other example in which Alice goes on the Sixth Avenue every 24 hours for the whole week, instead Bob every 24 hours from Monday to Friday. In this case, the routine of Bob will be modelled as intervals of 24 and 72 hours. So, if we consider a driver moving on Saturday, we would not be able to identify the driver, neither Alice nor Bob, since the interval is set to 24 hours. On the contrary, if the day of the week is taken into account, Alice will be correctly identified.

A. SR Algorithm

Let us consider a target vehicle belonging to a driver d. The SR algorithm acts in four phases:

a) Model Generation Dataset: Whenever a vehicle is used, its sensors register pieces of information about several *features*, e.g., the water temperature, the speed, the brake pressure, and so on. We assume to take trace of all these data in combination with the timestamp in which each instance of data is generated. Data are taken from the OBD-II port by using an OBD-II interface [21]. Each instance of data is called *interaction* of the driver d with the vehicle and it is denoted as $in_{i,d}$ where i is the timestamp. Interactions are composed by the timestamp, recorded with the following template: (day, month, year, hour, minute, second and day of the week) plus the others features obtained from the OBD-II.

b) FS paradigm: To mitigate the possible over-fitting error, we implement the *FSParadigm* (Figure 2).

FSParadigm is designed to select the best features to use. It firstly ranks all features applying the *Gain Ratio* approach and then features are sorted in ascending order. Those features with rank equal to zero are discarded. Then, the average-rank among all features not correlated to the timestamp is calculated. The FS discards those features, except those related to time, whose rank sum is less than or equal to the average-rank.

c) Model Generation Algorithm: Let us consider that a vehicle may be driven by d but also by other people, e.g., friends or relatives of d. In the modelling phase, our algorithm

(Figure 3) considers all the past interactions recorded by the vehicle and labels with 1 each interaction that belongs to d, 0 otherwise. The labelled interactions are sent to a DT algorithm that generates the model for the driver d.

In particular, in line 5, *FSParadigm* is the Feature Selection paradigm we described above as part of SR and line 6 (*MLAlgorithm*) indicates the ML algorithm in use with the subset of features obtained before.

d) SR Identification strategy: Once the model is generated, SR makes the identification evaluating each interaction. In particular, SR links an interaction to the vehicles' owner if the ML algorithm predicts and labels it as 1, otherwise 0.

V. Secure Routine Evaluation

We evaluated Secure Routine in two steps: first, we run it using two ML algorithms and we verified which of them best performs to identify drivers. Then, we compared Secure Routine with the following research works present in literature:

- Martinelli et al. [14] referred in the following as M.
- Kwak et al. [12] referred in the following as K.
- Girma et al. [9] referred in the following as G.

A. Datasets

We run the experiments using two datasets presented in [10], referred as Θ, and [22], referred as Ψ. The former is a dataset used also by M, K and G in their research works. So we can fairly make a comparison. However, the Θ dataset does not contain a fundamental feature used by SR, this is the *timestamp* of each represented instance. Nevertheless, Θ dataset contains the *engine runtime* that provides the minutes to be used as timestamp needed for SR to work.

On the other hand, Ψ dataset contains a timestamp for each instance by default. This feature allows Secure Routine to fully work by using all available pieces of information. In particular, SR expands the timestamp to generate all time dependent features. As far as we know, the other compared research works do not make use of this dataset to evaluate their proposal. So, to evaluate SR even in this case, we were able to re-run the work proposed by Martinelli et al. and calculate the results for the owner-driver identification. On the other side, the works K and G did not calculate the owner-driver identification and, also, it was not possible to re-run their algorithms since the implementation is not publicly available. In the specific case of G, the authors published only the pre-built model and we were not able to use it.

B. Metrics

To get a comparable result of SR with M, K and G, we evaluate *Accuracy* [23], *Precision* and *Recall* [14].

- *Accuracy* represents how often the model is making a correct prediction. It is the ratio between the number of correct predictions and the number of predictions:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

  where:
  - TP (True Positive) is the number of instances belonging to the vehicle's owner that are correctly predicted;
  - TN (True Negative) is the number of instances not belonging to the vehicle's owner that are correctly predicted;
  - FP (False Positive) is the number of instances belonging to another person but incorrectly predicted;
  - FN (False Negative) is the number of instances belonging to the vehicle's owner but incorrectly predicted.
- *Precision* measures how often the predicted instances belonging to the vehicle's owner are true. It is calculated as the ratio between TP and $TP + FP$:

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

- *Recall* identifies how often the instances belonging to vehicle's owner are correctly predicted. It is calculated as the ratio between TP and $TP + FN$:

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

To better estimate the three metrics depicted above, in our experiments we used the 10-fold cross-validation [24] approach. First, we split the dataset on 10 equal size subsets D_1, D_2, ..., D_{10}. Each instance of the dataset is randomly inserted in a subset. Then, we constructed 10 training sets Tr_1, Tr_2, ..., Tr_{10} and 10 testing sets Te_1, ..., Te_{10}. Tr_i is made of all subsets except D_i and Te_i is made of D_i

```
1   function FSParadigm(instances)
2       ranking ← GR(instances)
3       ranking_ordered ← order ranking ascending
4       features_>0 ← discard features with rank =
            0 from ranking_ordered
5       (features_no_timestamp_correlated,
            features_timestamp_correlated) ← features_>0
6       ranking_no_timestamp_correlated ← ranking from
            ranking_ordered of features present in
            features_no_timestamp_correlated
7       average_ranking ←
            mean(ranking_no_timestamp_correlated)
8       subset_no_timestamp_correlated ← discard
            features sum is less than or equal to
            the average_ranking from
            ranking_no_timestamp_correlated
9       subset ← subset_no_timestamp_correlated ∪
            features_timestamp_correlated
10      return subset
```

Figure 2. Feature Selection Paradigm

```
1   function generate_model(d)
2       ins_d ← get interactions from db made by d,
            labeling 1
3       ins_o ← get interactions from db made by
            others, labeling 0
4       ins_all ← ins_d ∪ ins_o
5       subset ← FSParadigm(ins_all)
6       model ← MLAlgorithm(ins_all with features
            from subset)
7       return model
```

Figure 3. Secure Routine Model Generation

with $i \in \{1, 2, ..., 10\}$. For each pair (Tr_i, Te_i) is calculated $Accuracy_i$, $Precision_i$ and $Recall_i$. Finally, we calculated the final value of $Accuracy$, $Precision$ and $Recall$ as the mean of $Accuracy_i$, $Precision_i$ and $Recall_i$, respectively.

C. Experiments

We performed four types of experiments to evaluate Secure Routine. The first experiment is related to multi-driver identification problem [14], i.e., properly identify who is the driver. However, as step zero, we decided to find the most suitable ML algorithm with the best features set to evaluate SR. We leverage on Weka [25] as software that contains a collection of visualization tools and algorithms for data analysis and predictive modelling. So, we used the available Gain Ratio method to rank each feature. Then, we employed *J48*, which is the implementation of the C4.5 algorithm, and RF algorithm over the driver identification.

In this step, results are obtained on Θ dataset. It contains data from 10 drivers. Figure 4(a) shows the driver instances' distribution. Drivers have 9438 instances on average: Driver 4 has the highest number of instances with 13244 samples while Driver 1 has the lowest number with 7240 instances. In addition, drivers drove two times in the same path in similar time-window. Dataset instances are recorded per second.

Table I shows the results obtained comparing SR implemented into J48 and RF algorithms applied to the driver identification using Θ dataset. RF algorithm with feature selection (37 features) obtained the best Precision and Recall.

After selecting SR with RF and the most appropriate features ranked by the Gain Ratio method, we show the first experiment results obtained by comparing SR with the work in M, K and G on the Θ dataset. As shown in Table II(a), Secure Routine and K achieves the best results. Note that M did not calculate the accuracy in the paper, so we established this value through the replication of their experiment. Instead, K did not provide on their research Precision and Recall. Finally, for G we were not able to retrieve the exact Accuracy.

As we can see in Table II(b), SR achieves almost a perfect Precision, i.e., 100%, but with the worst Recall and this depends on the features selection. In fact, if we increment the number of features, we increase the Recall but the Precision is decreased. Here, we decided to obtain a higher Precision selecting the most appropriate features using the Gain Ratio.

The second experiment is related to the *Owner Driver identification*, i.e., does the instance belong to the vehicle's owner? In this case, we compared SR only with M since K and G did not calculate the owner driver identification. As stated by the authors of M, they use the same feature set for both the multi-driver and owner driver identification.

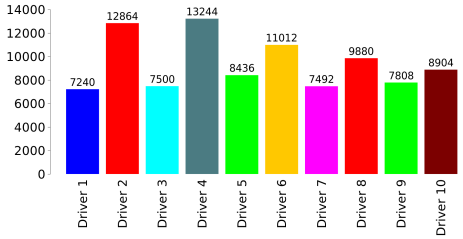
The third experiment that we propose is related to the multi-driver identification on the Ψ dataset. This contains data from 14 drivers. Figure 4(b) shows that drivers' instances are not equally distributed. For example, Driver 1 has the highest number of instances with 13617 samples, whereas Driver 10 has the lowest number with only 7 instances. This may depend on the fact that some users drive frequently whereas other users rarely. However, 7 instances are not enough to build a model for the Driver 10. So, we decided to exclude Driver 10 instances in our experiments to not alter the final result. Also, many instances contain empty values because of errors on gathering data. Instances are recorded every 7 seconds.

Compared to the Θ dataset, Ψ contains by default 32 features. Nevertheless, five of these features are timestamp related and are *minute*, *hour*, *day of the week*, *month*, *year*. Other features, such as, *model*, *car_year*, are removed since they do not give any useful information about the user driving style. The dataset also contains *engine_runtime* from which we extract *engine_runtime_minute*.

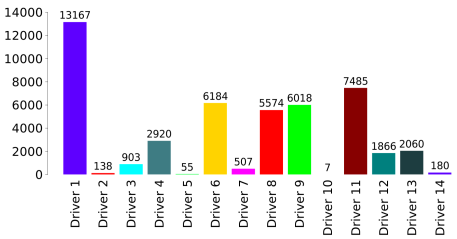
In this experiment, we used the GR method for features selection. Starting from pruned Ψ dataset, we evaluated SR. As previously stated, we know that there are no other research works that use this dataset. So, we had only the possibility to replicate the best solution proposed by M.

Table II(c) shows that Secure Routine with feature selection achieves the best result both for Precision and Recall.

To conclude the evaluation, last experiment focused on the owner driver identification. Table II(d) indicates that SR with features selection has the best performance when compared with M. SR obtained an average precision of 99,6%, which means that for 8 drivers SR established a perfect Precision whereas M achieved this Precision only for 4 drivers with an average Precision of 95,1%. Regarding the Recall, SR largely



(a) Number of instances for each driver in Θ



(b) Number of instances for each driver in Ψ

Figure 4. Driver distributions on the datasets.

TABLE I. COMPARING SR USING J48 AND RANDOM FOREST OVER THE MULTI-DRIVER IDENTIFICATION PROBLEM.

J48				Random Forest			
All features		Feature selection		All features		Feature selection	
Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
99,2%	99,2%	99,3%	99,3%	99,3%	99,3%	99,6%	99,6%

TABLE II. COMPARISON OF SECURE ROUTINE WITH RELATED WORKS

(a) Comparison of Secure Routine with M, K and G.

Secure Routine		M		K		G	
Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
99,6%	99,6%	99,2%	99,2%	N.A.	N.A.	98,8%	98,1%
Accuracy		Accuracy		Accuracy		Accuracy	
99,6%		99,2%		99,6%		N.A.	

(b) Comparison of Secure Routine with M.

Secure Routine		M	
Avg. Precision	Avg. Recall	Avg. Precision	Avg. Recall
99,8%	98,5%	99,3%	99,3%

(c) Comparison of Secure Routine with M for multi-driver identification.

Secure Routine		M	
Precision	Recall	Precision	Recall
99,4%	99,4%	90,4%	89,8%

(d) Comparison of Secure Routine with M for owner identification.

Secure Routine		M	
Avg. Precision	Avg. Recall	Avg. Precision	Avg. Recall
99,6%	98,1%	95,1%	82,9%

outperformed M in percentage and SR achieved a perfect Recall score for one driver, whereas M never obtained a perfect Recall.

VI. CONCLUSION AND FUTURE WORK

In this paper, we introduced for the first time the Secure Routine paradigm to identify the vehicle's owner taking into account the driving style. Also, we presented the algorithm implemented by means of machine learning algorithms and we showed how SR works to identify the driver. Then, we compared SR with other three existing research papers and we evaluated them considering Precision, Accuracy and Recall metrics. Experiments made use of two different datasets. Findings showed that SR obtains the best results compared with the other algorithms considering both experiments regarding the identification of the vehicle's owner and the multi-driver.

As future work, we plan to improve the algorithm of Secure Routine by considering additional features to increase its identification capabilities, i.e., statistical features. We will also improve our *FSParadigm* to enable a better feature selection.

ACKNOWLEDGMENT

This work has been partially supported by the COSCA research project (NGI_TRUST 2nd Open Call 2019002).

REFERENCES

[1] B. Johnston, "Rivervale reveal DVLA data to uncover the most stolen cars in the UK," 02 2020, URL:https://www.rivervaleleasing.co.uk/blog/posts/most-stolen-cars-uk-theft#sthash.iCWvAg4d.dpuf [retrieved: 09, 2020].

[2] Y. Xiong and H. Lin, "Routine based analysis for user classification and location prediction," in 2012 9th International Conference on Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing, Sep. 2012, pp. 96–103.

[3] I. Lavie, A. Steiner, and A. Sfard, "Routines we live by: from ritual to exploration," Educational Studies in Mathematics, vol. 101, no. 2, Jun 2019, pp. 153–176, URL:https://doi.org/10.1007/s10649-018-9817-4.

[4] N. Banovic, T. Buzali, F. Chevalier, J. Mankoff, and A. K. Dey, "Modeling and understanding human routine behavior," in Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, ser. CHI '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 248–260, URL:https://doi.org/10.1145/2858036.2858557.

[5] "The OBDII Home Page", "Obd-ii background," URL:http://www.obdii.com/background.html [retrieved: 09, 2020].

[6] M. Bernardi, M. Cimitile, F. Martinelli, and F. Mercaldo, "Driver and path detection through time-series classification," Journal of Advanced Transportation, vol. 2018, 03 2018, pp. 1–20.

[7] Z. Gao, L. Li, J. Feng, R. Yu, X. Wang, and C. Yin, "Driver identification based on stop-and-go events using naturalistic driving data," in 2018 11th International Symposium on Computational Intelligence and Design (ISCID), vol. 01, Dec 2018, pp. 306–310.

[8] B. Wang, S. Panigrahi, M. Narsude, and A. Mohanty, "Driver identification using vehicle telematics data," in SAE Technical Paper. SAE International, 03 2017. [Online]. Available: https://doi.org/10.4271/2017-01-1372

[9] A. Girma, X. Yan, and A. Homaifar, "Driver identification based on vehicle telematics data using lstm-recurrent neural network," in 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI), 2019, pp. 894–902.

[10] HCRL, "Driving dataset," URL:http://ocslab.hksecurity.net/Datasets/driving-dataset [retrieved: 09, 2020].

[11] P. Rettore, "Vehicular traces," 2018, URL:http://www.rettore.com.br/prof/vehicular-trace/ [retrieved: 09, 2020].

[12] B.-I. Kwak, J. Woo, and H. K. Kim, "Know your master: Driver profiling-based anti-theft method," in PST 2016, 12 2016, pp. 211–218.

[13] F. Martinelli, F. Mercaldo, V. Nardone, A. Orlando, and A. Santone, "Who's driving my car? a machine learning based approach to driver identification," 01 2018, pp. 367–372.

[14] F. Martinelli, F. Mercaldo, A. Orlando, V. Nardone, A. Santone, and A. K. Sangaiah, "Human behavior characterization for driving style recognition in vehicle system," Computers & Electrical Engineering, vol. 83, 2020, p. 102504. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0045790617329531

[15] J. R. Quinlan, C4.5: Programs for Machine Learning. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[16] C. E. Shannon, "A mathematical theory of communication," Bell System Technical Journal, vol. 27, no. 3, 1948, pp. 379–423. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/j.1538-7305.1948.tb01338.x

[17] A. Cutler, D. R. Cutler, and J. R. Stevens, Random Forests. Boston, MA: Springer US, 2012, pp. 157–175. [Online]. Available: https://doi.org/10.1007/978-1-4419-9326-7_5

[18] L. Breiman, "Random forests," Machine Learning, vol. 45, no. 1, 10 2001, pp. 5–32. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[19] G. M. Hodgson, "The ubiquity of habits and rules," Cambridge Journal of Economics, vol. 21, no. 6, 11 1997, pp. 663–684, URL:https://doi.org/10.1093/oxfordjournals.cje.a013692.

[20] E. Shi, Y. Niu, M. Jakobsson, and R. Chow, "Implicit authentication through learning user behavior," in Information Security, M. Burmester, G. Tsudik, S. Magliveras, and I. Ilić, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 99–113.

[21] "Elm Electronics Inc", "Elm327 obd to rs232 interpreter," 2017, URL:https://www.elmelectronics.com/wp-content/uploads/2016/07/ELM327DS.pdf [retrieved: 09, 2020].

[22] C. A. d. S. Barreto, "OBDdatasets," 2018, URL:https://github.com/cephasax/OBDdatasets/blob/master/masterDegreeResearch/dailyRoutes.csv [retrieved: 09, 2020].

[23] J. Torres, First Contact with Deep Learning, practical introduction with Keras. Watch this space, 7 2018, URL:https://torres.ai/first-contact-deep-learning-practical-introduction-keras/ [retrieved: 09, 2020].

[24] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2, ser. IJCAI'95. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, p. 1137–1143.

[25] I. Witten, M. Hall, E. Frank, G. Holmes, B. Pfahringer, and P. Reutemann, "The weka data mining software: An update," SIGKDD Explorations, vol. 11, 11 2009, pp. 10–18.

 ISBN: 978-1-61208-795-5