Driver Behavior Detection Using Smartphone Signals

Aviv Gelfand, Aviad Elron, Amit Chen

In this project we will implement and investigate a method called the "D3: Abnormal driving behaviors detection and identification using smartphone sensors"[1].The method applies a support vector machine classification algorithm for the detection of dangerous driving behavior based on the sensors of smartphones.
We will try to implement the algorithm on a different driving behavior data set from other sensors described in the article, we will describe what adjustments we made to the data and the algorithm, and finally we will describe the experiments, successful as far as we understand, that we conducted in the process. The following figure describes the general workflow of the method we apply in this project.

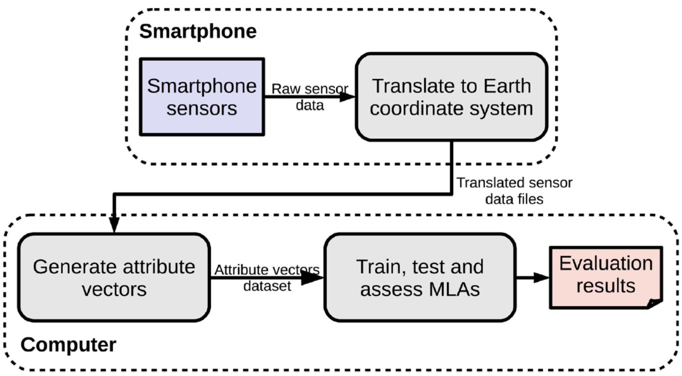


Fig 1. High level view of our evaluation pipeline showing processing steps from raw sensor data sampling to training, testing, and assessing MLAs. [2]

In addition to the required submission format, this project is also documented in the following GitHub repository. We hope this paper will be a milestone in a broader project that will develop an application to indicate and alert risky bus driver's behaviors.

The Data In the Article

In the article, the data represent driving traces from the real-world environment, which means, traces of human drivers that were collected over 6 months. The data was collected from smartphone sensors, and included two main features: *acceleration* (the rate of change of velocity) and *orientation* (location and direction in space). Note the authors did not specify the specific units measured for each feature.

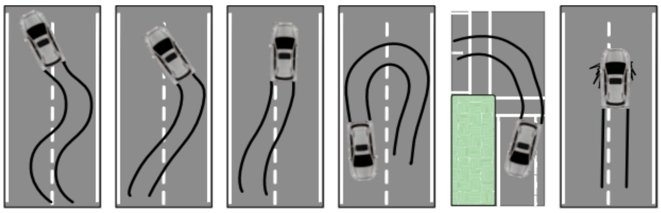
---

[1]Z. Chen, J. Yu, Y. Zhu, Y. Chen and M. Li, "D3: Abnormal driving behaviors detection and identification using smartphone sensors," 2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), Seattle, WA, USA, 2015, pp. 524-532, doi:

10.1109/SAHCN.2015.7338354.

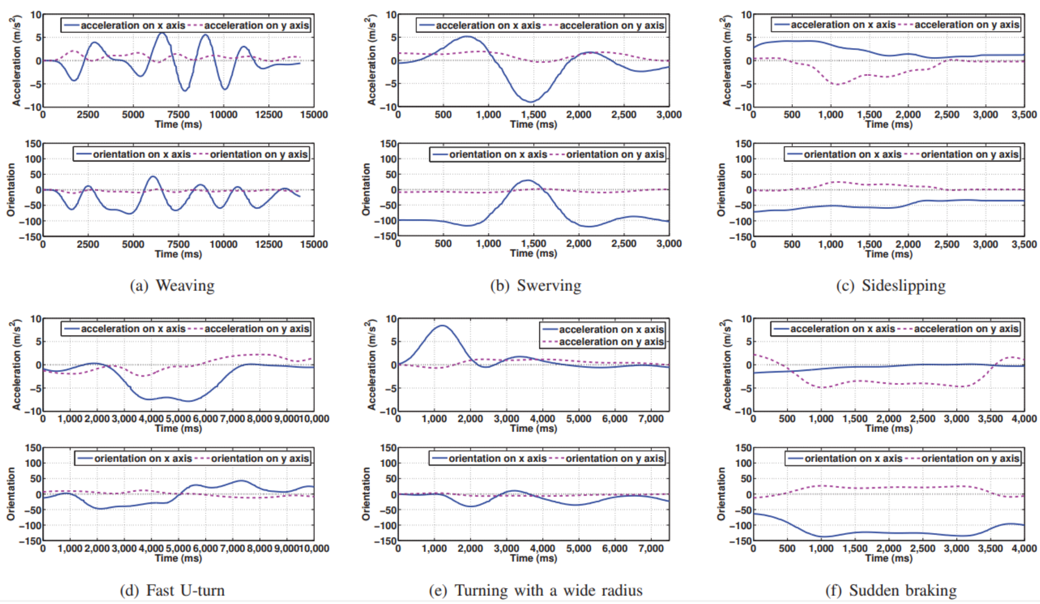
[2] Figure is taken from Ferreira J Júnior, Carvalho E, Ferreira BV, de Souza C, Suhara Y, Pentland A, Pessin G. Driver behavior profiling: An investigation with different smartphone sensors and machine learning. PLoS One. 2017 Apr 10;12(4):e0174959. doi: 10.1371/journal.pone.0174959. PMID: 28394925; PMCID: PMC5386255.

The following figures from the article show the acceleration and orientation patterns of different risky driving behaviors. For example, when a vehicle brakes suddenly (image f in the following figure), acceleration_x remains flat while acceleration_y sharply downs and keeps negative for some time. Thus, the standard deviation and value range of acceleration_x are small. On acceleration_y, the standard deviation is large at the beginning and ending of a sudden braking, and the range of acceleration_y is large. Moreover, there are no obvious changes in both oriantation_x and oriantation_y. Since sudden braking is an abrupt driving behavior, the duration is short.



(a) (b) (c) (d) (e) (f)

Fig. 1: Six types of abnormal driving behaviors: (a) Weaving, (b) Swerving, (c) Sideslipping, (d) Fast U-turn, (e) Turning with a wide radius, (f) Sudden braking.



The Method In the Article - part 1: Feature extraction

The authors demonstrate this procedure by extracting unique features from readings of smartphones' accelerometers and orientation sensors. They first identify the representative features to capture the driving behavior patterns (table 1). The image on the left (fig 3.) illustrates the sliding window process that is performed to extract features from the serial data of the signals.

A *sliding window* approach is then used for the

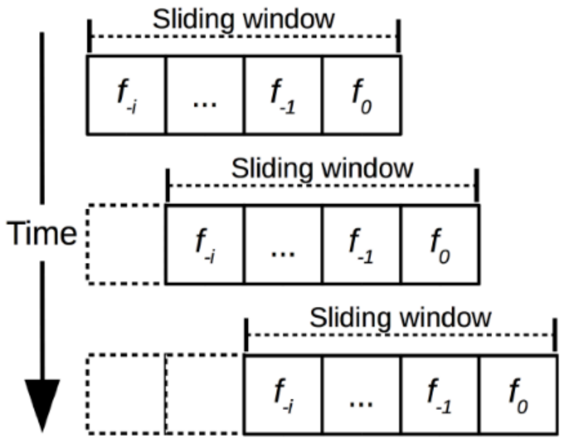


Fig 3. Time window composed of *nf* one-second frames which group raw sensor data samples. The time window slides in 1 frame increments as time passes. f_0 is the frame of the current second, f_{-1} is the frame of the previous second, and so forth down to f_{-i}, where $i = nf - 1$.

https://doi.org/10.1371/journal.pone.0174959.g003

TABLE I: Features Extracted

Feature	Description
$range_{acc,x}$	subtraction of maximum minus minimum value of acc_x
$range_{acc,y}$	subtraction of maximum minus minimum value of acc_y
$\sigma_{acc,x}$	standard deviation of acc_x
$\sigma_{acc,y}$	standard deviation of acc_y
$\sigma_{ori,x}$	standard deviation of ori_x
$\sigma_{ori,y}$	standard deviation of ori_y
$\mu_{acc,x}$	mean value of acc_x
$\mu_{acc,y}$	mean value of acc_y
$\mu_{ori,x}$	mean value of ori_x
$\mu_{ori,y}$	mean value of ori_y
$\mu_{acc,x,1}$	mean value of 1^{st} half of acc_x
$\mu_{acc,x,2}$	mean value of 2^{nd} half of acc_x
$max_{ori,x}$	maximum value of ori_x
$max_{ori,y}$	maximum value of ori_y
$min_{acc,y}$	minimum value of acc_y
t	time duration between the begining and the ending of a driving behavior

As shown in Figure 3, the sliding window method involves segmenting the continuous time-series data into overlapping windows of size t, where each window captures a subset of the data. For each of these windows, various statistical features are computed—these include measures of central tendency, variability, and extremal values (as detailed in Table 1). The number of rows in the new features matrix is dependent on the number of frames (nf) in the sliding window .

By applying this technique, the temporal structure of the data is preserved, allowing for a more nuanced understanding of the underlying patterns within the sensor data. This methodology not only facilitates a robust analysis of the temporal aspects of the sensor outputs but also enhances the detection and characterization of nuanced variations in driving behaviors over time.

The Method In the Article - part 2: Support Vector Machine Algorithm

A machine learning algorithm, Support Vector Machine (SVM), is employed to train the features and output a classifier model that conducts fine-grained identification. SVM seeks the best decision boundary which separates two classes with the highest generalization ability.

Support Vector Machines (SVMs) are a type of supervised machine learning algorithm primarily used for classification tasks, though they can also be employed in regression. The core idea behind SVM is to find the best boundary (hyperplane) that separates classes of data points in a high-dimensional space. This boundary is chosen to maximize the margin, which is the distance between the nearest points (support vectors) of each class and the hyperplane itself. By doing so, SVM aims to improve the model's generalizability and robustness, minimizing the risk of misclassifying new examples.

$$Margin = \min_{i=1,2,\ldots,N} dist_{x_i} = \min_{i=1,2,\ldots,N} \left| \frac{(\vec{x}_i * \vec{W} + b)}{\|\vec{W}\|} \right|$$

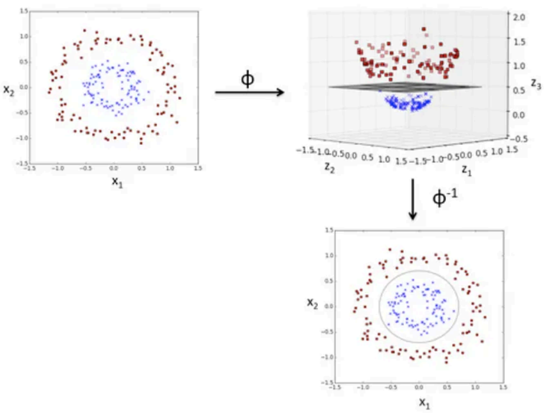
$$If\ we\ \boldsymbol{define}\ \min_{i=1,2,\dots,N}\left|(\vec{x}_i * \vec{W} + b)\right| = 1, then\ Margin = \frac{1}{\|W\|}\ and\ \left|(\vec{x}_i * \vec{W} + b)\right| \geq 1 \rightarrow$$
$$y_i(\vec{x}_i * \vec{W} + b) \geq 1\ since\ y \in \{1, -1\}$$

$$We\ want\ to\ \boldsymbol{maximize}\ \frac{\mathbf{1}}{\|\boldsymbol{W}\|}\ \boldsymbol{subject\ to}: \boldsymbol{y_i(\vec{x}_i * \vec{W} + b) - 1 \geq 0\ \forall i \in x}$$

In cases where data is not linearly separable, SVM uses a technique called the kernel trick. This approach involves mapping data to a higher-dimensional space where a hyperplane can effectively do the separation.

$$\Phi(\mathrm{x}) \rightarrow x'\ (mapping\ x\ to\ another\ point\ x'\ in\ the\ new\ sapce)$$

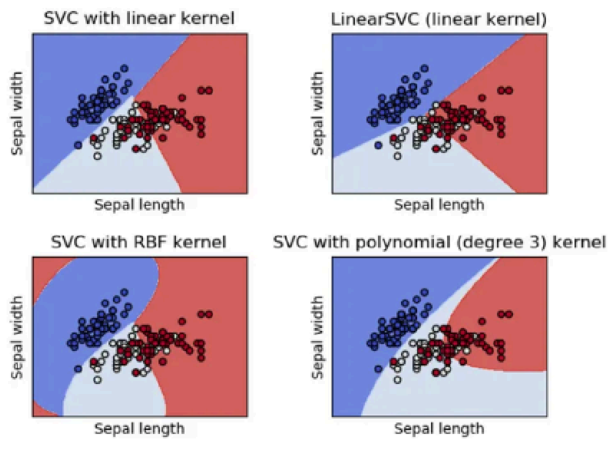
After we've done classification in the transformed space, we can map the hyperplane back, and the boundaries may become nonlinear



Commonly used kernels include polynomial, radial basis function (RBF), and sigmoid.

$$K(\mathbf{x},\mathbf{y}) = (\mathbf{x}\cdot\mathbf{y}+1)^p$$
$$K(\mathbf{x},\mathbf{y}) = \exp\left\{-\|\mathbf{x}-\mathbf{y}\|^2 / 2\sigma^2\right\}$$
$$K(\mathbf{x},\mathbf{y}) = \tanh(\kappa\mathbf{x}\cdot\mathbf{y}-\delta)$$

1st is polynomial (includes x•x as special case
2nd is radial basis function (gaussians)
3rd is sigmoid (neural net activation function)



Decision boundary obtained with kernel methods, 3 classes, from [Scikit-Learn](#)

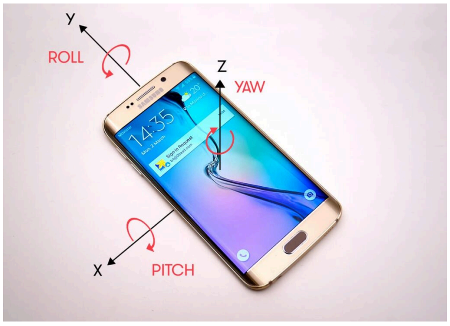
Same Approach, Different Data - About Our Data Set

In this project we will explore a the [Phone sensor data while driving a car and normal or aggressive driving behavior classification](#)[3]. It is a public dataset for driver behavior

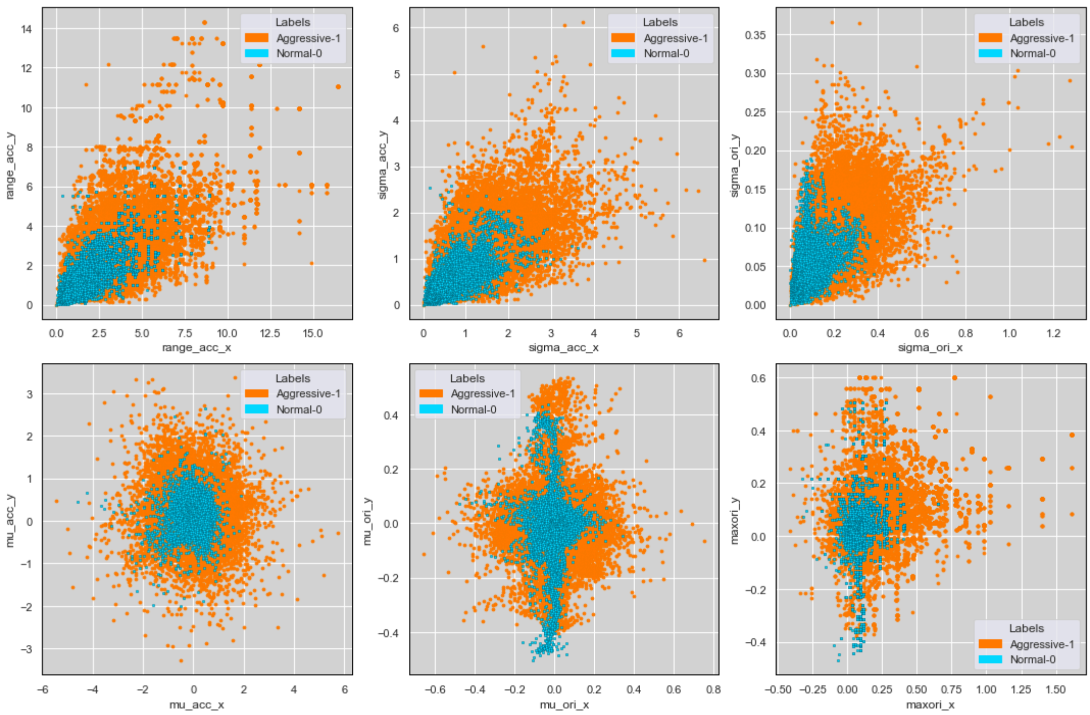
[3] S. Nazirkar, "Phone sensor data while driving a car and normal or aggressive driving behavior classification". Mendeley, 2021. doi: 10.17632/5STN873WFT.1.

classification. The data has been recorded on an android phone attached to the dashboard of the car. Data was collected while driving the car on city roads in mild traffic.
The raw features recorded are : Longitude, Latitude, Speed, Distance, Time, Acc X, Acc Y, Acc Z, Heading, gyro_x, gyro_y, gyro_z (Acc - Accelerometer X,Y,Z axis in meters per second squared (m/s2), Gyro - Gyroscope X,Y, Z axis in degrees per second (°/s)).
There are no orientation features provided recordings in our data set. This might oppose a challenge to the success of the algorithm in our case study article.

Orientation Vs Gyroscope Measurements
In the context of inertial sensors used in smartphones and other mobile devices, gyroscopes and accelerometers provide complementary data that can be used to determine the device's orientation in space. The gyroscope measures angular velocity, while the accelerometer measures linear acceleration. Understanding the difference between the gyroscope's measurements and orientation angles such as pitch and roll is crucial for applications ranging from navigation to gaming and augmented reality.
Pitch and roll are two of the three orientation angles (the third being yaw) that describe the device's orientation relative to a stable reference such as the Earth. They are typically derived using sensor fusion algorithms that combine data from both the accelerometer and gyroscope. We instead, decided to train the model using the gyroscope data as columns, and the results were satisfying.



Experiments and Results
We then applied the above mentioned sliding window approach to extract the features with window sizes ranging between 2 to 7 frames.
Before choosing the model itself we decided to test our features on several scatter graphs in order to see the distribution of the data and how effective a linear separator would be:



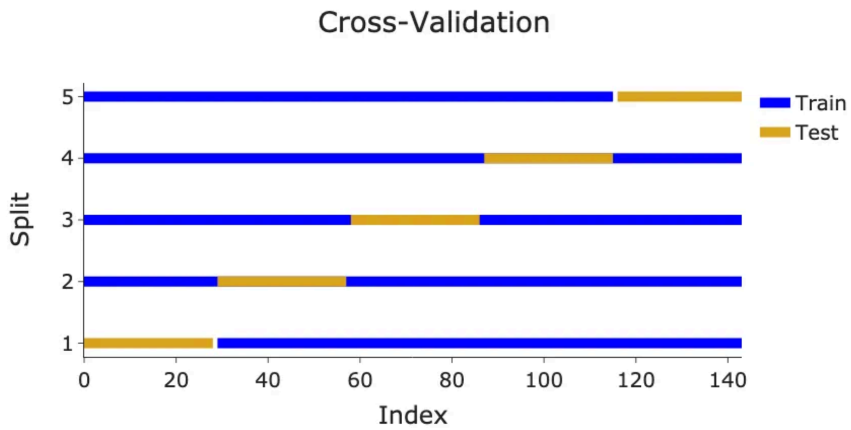
There is some overlap between the samples, with the samples between normal driving (0) delimited by several dense groups in each diagram, while for aggressive driving (1) they are also in the same area but also scattered over the entire diagram, and have a noticeable presence for more extreme values. This foreshadows a characteristic of samples for which non-linear separation is more suitable.

Our model fitting method began with a train-test split to indices into a training and a test group, so that the split fits the serial-temporal structure of the data. We extended this method using Cross Validation as described below.
Scale the training set and then scale the test set in light of the training set (to avoid data leakage).

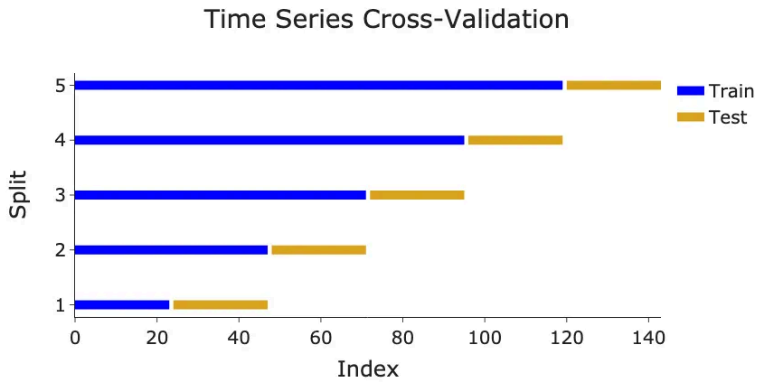
We then ran a comparison between the following models to select the best kernel: linear kernel, logistic kernel, polynomial and RBF (Radial Basis Function). For each model, we also tested a weighted version designed to penalize according to scale weights in relation to the relative weight of each label. We wanted to avoid prediction problems that could arise from an unbalanced sample like in our own data.

Cross Validation and Time Series Cross Validation
Cross-validation is a widely employed technique for assessing the performance and generalization capability of predictive models, as well as for selecting the optimal model parameters. The most rudimentary and commonly utilized approach is the classic train-test split paradigm. In this method, the available data is partitioned into two subsets: a training set, which is used to estimate the model parameters, and a test set, which is employed to evaluate the model's predictive performance on unseen data.

The concept of cross-validation extends this idea by executing the train-test split process multiple times, varying the composition of the training and test sets across iterations. This approach aims to leverage every available data point for both training and evaluation purposes, thereby enhancing the robustness and reliability of the model selection process and ensuring the identification of the most generalizable model across the entire data distribution.



However, it is important to note that the aforementioned cross-validation strategy is not an effective or valid approach for forecasting models that exhibit temporal dependencies. In the context of time series data, the objective is to predict future values based on historical observations. Consequently, the conventional cross-validation approach, where the training set may contain data points temporally subsequent to those in the test set, would result in an undesirable phenomenon known as data leakage, which should be stringently avoided.
To circumvent this issue and ensure the validity of the forecasting model evaluation, it is imperative to adhere to a specific cross-validation procedure that preserves the temporal ordering of the data. Specifically, the test set must consistently comprise data points with higher temporal indices (typically representing later time periods) than those in the training set. This approach ensures that the model is trained exclusively on historical data and evaluated on its ability to forecast future values, thereby accurately reflecting the real-world application of forecasting models.

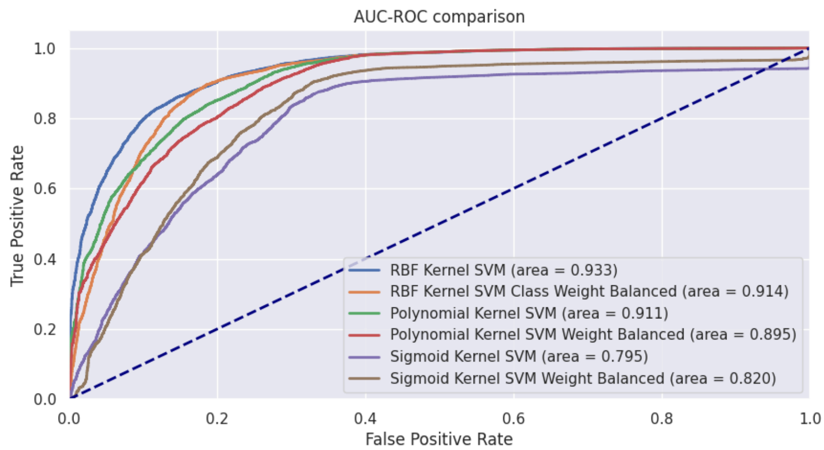


For each model we trained, we used the cross-validation method and calculated the average score as shown in the following example:

split	fit_time	score_time	test_accuracy	test_precision	test_mcc
1	15.022048	2.303058	0.836009	0.86018	0.615489
2	58.352187	4.373766	0.852753	0.859231	0.653573
3	134.929158	6.470274	0.857382	0.86762	0.665425
4	248.837985	8.318508	0.858859	0.872172	0.669543
5	414.820011	10.053894	0.860238	0.878188	0.673881
avg	**174.392278**	**6.3039**	**0.853048**	**0.867478**	**0.655582**

Performance Evaluation And Results

We used several metrics to test out our model performance:
accuracy scores, that is, the percentage of correct observations out of all predicted observations. The index itself is the simplest but also not the most effective, especially not in an unbalanced sample like ours, but we preferred to keep it anyway.
The *AUC* index, a supplementary index to the error matrix and used in the article on which we reference to. The index tests the performance of a classification model and by using the ratio between the rate of true charges and the rate of false charges. Although there is criticism of the index, since it was based on it in the article, we made comparisons with it as well.
In addition, we plot the Receiver Operating Characteristic (ROC) curve. The ROC plot is a graphical representation that illustrates the performance of a binary classification model by plotting the true positive rate (sensitivity) against the false positive rate (1 - specificity) at various classification thresholds. The AUC, ranging from 0 to 1, measures the area under this ROC curve and serves as a comprehensive metric for evaluating the model's ability to discriminate between the two classes.



Among the models compared, the RBF Kernel SVM (Radial Basis Function Kernel Support Vector Machine) achieves the highest AUC of 0.933, indicating the best overall classification performance.

MCC index (Matthews correlation coefficient) - The Matthews Correlation Coefficient (MCC) is a comprehensive performance metric that incorporates all elements of the confusion matrix, effectively summarizing the predictive power of a binary classification model in a single value. The MCC index ranges from -1 to 1, where -1 represents a completely reversed prediction (i.e., positive instances predicted as negative and vice versa), 0 indicates a random prediction, and 1 signifies a perfect prediction.

$$\mathrm{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

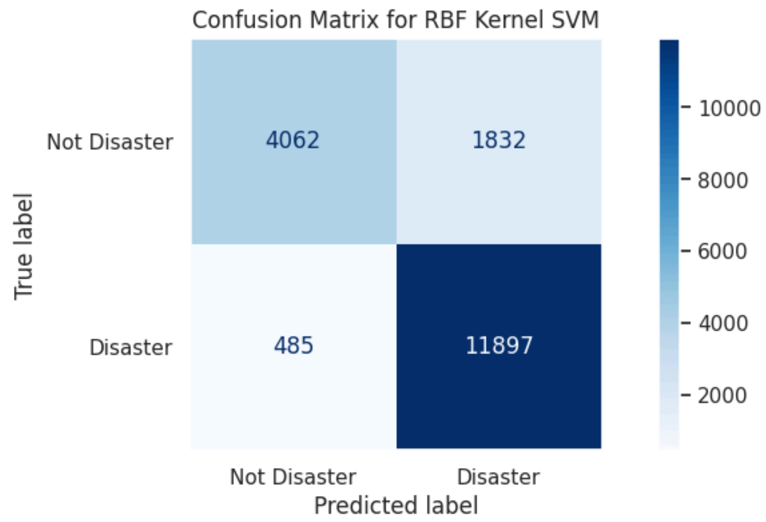
A notable advantage of the MCC over other metrics, such as accuracy, is its ability to account for class imbalance in the data. In situations where the class distribution is skewed, with one class significantly outnumbering the other, accuracy can be a misleading metric, as a naive classifier that consistently predicts the majority class may achieve a deceptively high accuracy score. The MCC, on the other hand, provides a more comprehensive and balanced assessment by considering the true positive, true negative, false positive, and false negative rates, effectively accounting for the impact of class imbalance.

The following table summarizes the metrics achieved by each model we compare.

Model Name	**MCC**	**Accuracy**	**Auc-Roc**	**TP**	**FP**	**TN**	**FN**	**train_time_secs**
RBF Kernel SVM	0.702806	0.873	0.933	4062	485	11897	1832	271.53
Polynomial Kernel SVM	0.680089	0.863	0.911	3784	388	11994	2110	282.643
RBF Kernel SVM Class Weight Balanced	0.660421	0.853	0.914	3426	212	12170	2468	291.895
Linear SVM Weight Balanced	0.649615	0.852	NaN	3978	797	11585	1916	0.193
Polynomial Kernel SVM Weight Balanced	0.648664	0.848	0.895	3326	204	12178	2568	336.197
Linear SVM	0.607589	0.826	NaN	4467	1757	10625	1427	6.483
Sigmoid Kernel SVM Weight Balanced	0.597455	0.83	0.82	3725	930	11452	2169	231.656
Sigmoid Kernel SVM	0.547395	0.805	0.795	3929	1595	10787	1965	236.225

Overall, the RBF Kernel SVM model emerges as the top-performing model based on the reported metrics, followed by the Polynomial Kernel SVM and RBF Kernel SVM Class Weight Balanced models. The Sigmoid Kernel SVM models underperform compared to the others. Weight balancing techniques seem to have a mixed impact on performance, improving some models while slightly degrading others.

In binary classification problems, one class is referred to as "positive" and the other as "negative". There are four possible outcomes: true positive (correct positive prediction), false positive (incorrect positive prediction), true negative (correct negative prediction), and false negative (incorrect negative prediction). These values are typically presented in a confusion matrix, which we utilized to evaluate the performance of our models. We attach the confusion matrix of our final model:



What next?

The next step in Fruitech will be complex and interesting. We are interested in continuing the adaptation of the model to the data with another difference, observations recorded during bus trips from our mobile phone. We may also consider training other types of models such as Random Forest or a neural network. This journey has only just begun, and is already very educational and exciting for us.