**How these hackathon winners apply Machine Learning to minimize rash driving**

Hackathons have become the go-to method for developers and companies to innovate in tech and build great products in a short span of time. With tens of thousands of developers participating in hackathons across the globe every month, it’s a great way to collaborate and work on solving real-life issues using tech.

“Along with being stimulating and productive, hackathons are fun” – says Team Vikings, who won the first prize (a brand new Harley-Davidson bike!) in the recently concluded [GO-HACK hackathon](https://go-hack.hackerearth.com/). The team built Rashalytics – a comprehensive platform for analysing and minimising rash driving. And now, they have big plans of taking this hack live for the public.

**What is Rashalytics?**

Rashalytics is a system that promises to mitigate the problem of rash driving by intelligently incentivising or penalising the driver based on his driving style. It has been designed to reduce the number of accidents that have increased with the hyperlocal on-demand delivery requiring breakneck speeds of the various products.

The system is able to extract the data of rich metrics like sharp acceleration, hard braking, sharp turns, etc. from the driver’s android phone, which is used to train the machine learning models.

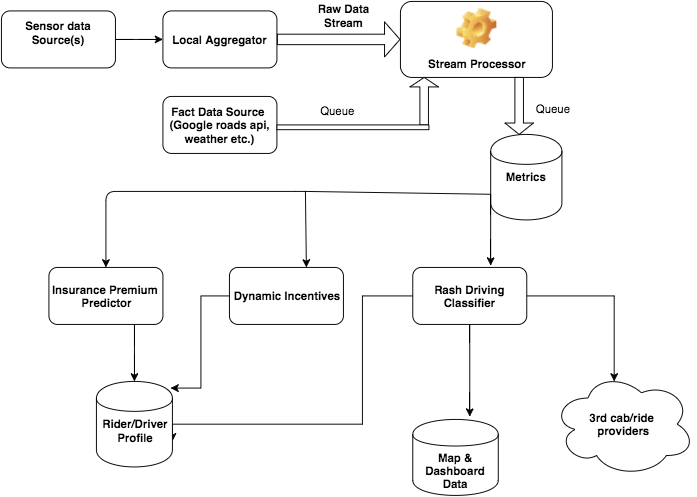
**Technologies/platforms/languages**

* **Nodejs** – To create the API server and the mock sensor data generator
* **Kafka** – To build the data pipeline
* **Apache Spark** – To process the real-time data stream and generate metrics to measure the driving quality
* **ReactJs** – To create the dashboard web app
* **Google roads & maps API** : To get the traffic and ETA data

**Functionality**

The system primarily consists of 4 parts:.

1. **The Android app:** Simulated by the team, this app aggregates locally and sends the chunks of sensor data to the API server via an HTTP endpoint.
2. **API Server:** This matches the data with the schema and if valid, it puts the data in Kafka queue.
3. **Engine:** Made with the Apache Spark, this helps sensor data to aggregate and form metrics such as sharp acceleration, hard braking, sharp turns, etc. These metrics, in turn, are used to generate a dynamic driving quality score for the driver. This score forms the basis of a lot of analytics and functionalities that this system provides.
4. **Dashboard:** The dashboard provides a beautiful and intuitive interface to take proactive decisions as well as run analytics using the provided APIs. It has been written using ReactJS.

Here’s the flow diagram showing how the whole system works:

This system allowed the team to create:

* A dynamic profile and the dashboard of the rider describing his driving style, which affects his rating.
* An actionable “real-time” rash driving reporting system which allows the authorities and the hub incharges to react before it’s too late.
* A dashboard usable by both the fleet managers and traffic police control board to visualise the data such as incident distribution by time, which tells at what time of the day a driver is more likely to drive in an unsafe manner.
* A modular system in which the new data sources, metrics, and models can be added so that the third-party vendors can be easily on-boarded onto the platform.

**Challenges**

Here are some of the challenges that the team faced while building this application:

1. Setting up the entire system architecture with different components by developing them in isolation and then combining them together to work seamlessly
2. Deciding the thresholds for different metrics after which the driving will be considered rash
3. Creating a linear predictor for the driving quality score vs time with only one data point
4. Creating a synthetic feature as generating the score itself is challenging enough