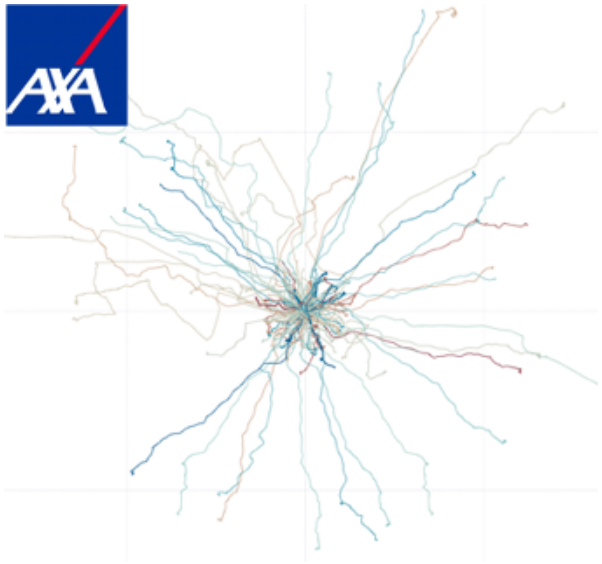
AXA Winners' Interview: Learning Telematic Fingerprints From GPS Data

Kaggle Team (http://blog.kaggle.com/author/kaggleteam/) | 04.20.2015

Team Driving It took second place (https://www.kaggle.com/c/axa-driver-telematics-analysis/leaderboard/private) in the hugely popular AXA Driver Telematics (https://www.kaggle.com/c/axa-driver-telematics-analysis) competition. This blog shares their experience working as a team to build a "telematic fingerprint" for drivers, making it possible to distinguish if a given driver was behind the wheel.

"***Our final telematics model consisted of an ensemble of 6 different models. We mainly used Random Forests, in combination with Gradient Boosting and Logistic Regression.***"

"Collaborating in a team is a great experience, we finished 2nd place because we joined forces. I definitely recommend it."



(http://blog.kaggle.com/wp-content/uploads/2015/04/Telematics_axa2.png)

1,528 TEAMS AND 1,865 PLAYERS COMPETED TO CREATE A "TELEMATIC FINGERPRINT" FOR DRIVERS.

What was your background prior to entering this challenge?

Scott: I work at an aerospace company as a mechanical engineer. I've done a fair amount of coding in Fortran, but have only been using python for a few months.

Janto: I'm a Cognitive Science MSc student. I'm interested in Computational Neuroscience and Machine Learning. My work involves modelling behavioural and brain data as well as applying machine learning. While I mostly work in MATLAB during the day I started coding in Python about 1.5 years ago and it has become my first choice for machine learning tasks (and nearly everything else). Currently, I'm working on my thesis at Oxford University.

Andrei: I work as a cloud architect at Sparked (http://sparked.nl (http://sparked.nl)), mainly focusing on bringing data analytics and machine learning to mainstream. In parallel, I work on finishing up my PhD thesis as a result of 4 years being a part of the Software Languages Team at the University of Koblenz-Landau in Germany.

How did you get started competing on Kaggle?

Scott: I got started in the Christmas Challenge, helping Santa's helper's. I wanted to use it as a method of learning python. I picked that challenge because it was more algorithm based than data analysis based.

Janto: This was my first competition on Kaggle. I had played with the Titanic and the MNIST data set to benchmark a few algorithms. My main motivation for joining Kaggle was to test how statistics and machine learning algorithms would perform on real datasets and learn about the strengths and weaknesses of different approaches, hands-on.

Andrei: Actually, this was my first competition. I like the automotive and related domains and decided to join.

What made you decide to enter this competition?

Janto: I liked the fact that the data was unlabeled which I found challenging. With only the raw GPS data given the competition left room for a lot of creative work in feature engineering, local testing and trip matching which became evident in the diversity of ideas on the forums. I expected the challenge to be quite different from the "supervised learning toolbox-battles".

Scott: I chose to do this analysis because it appeared to me that path matching would be a dominant method of ranking the routes. Since there is no out of the box path matching tools that I know of, it would be much more algorithm based and somewhere I could add value. I contrast this with data analysis techniques like feature generation, random forests, gradient boosted trees, etc. Which at the time I was not familiar with.

Andrei: The motivation was twofold. First, I enjoy things happening in the automotive and related fields. Second, I was planning to refresh my data science toolkit, that I used to apply in very special technical domains in the past. For this reason I was looking to join the team, since this is the best way to combine and distill the best approaches out there.

What preprocessing and supervised learning methods did you use?

Janto: I focused on the telematics modelling part and spent a great amount of time on feature engineering and data exploration at the beginning of the challenge. First I tried Self-Organizing Maps which didn't work well at the early stage when I only had few features.

For me, the crucial step was to turn the task into a supervised problem and the effort we put into ensembling telematics and trip matching.

We tackled the problem by combining the results of the trip matching that Scott had done and our driver signature models. The idea was to identify frequently taken trips as they were likely to be trips from the respective driver and apply supervised models to telematic features for unmatched trips.

Our final telematics model consisted of an ensemble of 6 different models. We mainly used Random Forests, in combination with Gradient Boosting and Logistic Regression. Some coarse grid searching I had done earlier in the competition yielded the model parameters and we didn't tune a lot in the end. We trained models on different partially independent feature sets each of us had extracted and combinations of those feature sets.

Random Forests worked pretty well and extremely fast: after decreasing the time spent for input/output routines by converting the data to .npy files, my first ensemble (a simple Random Forest combined with Logistic Regression) scored 0.86 on the leaderboard in about 30 minutes.

Scott: Like many people in the competition, we used a combination of path matching and a telematics approach. My focus was on path matching, and only switched to telematics at the end of the competition.

For the path matching, the most important preprocessing method was the RDP algorithm, which I discovered relatively early in my investigation phase when I stumbled upon this Stack Overflow post

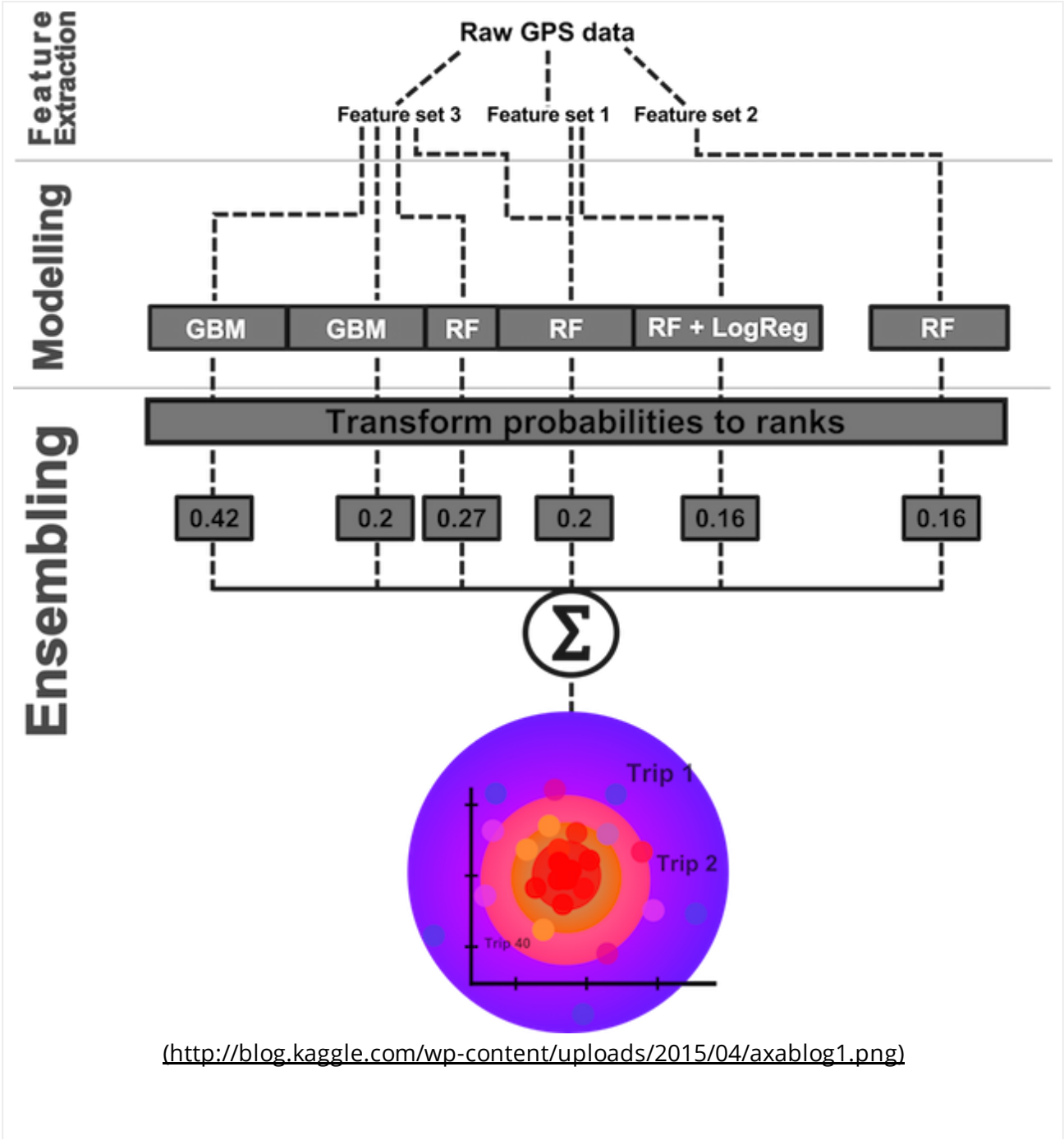
http://stackoverflow.com/questions/14631776/calculate-turning-points-pivot-points-in-trajectory-path (http://stackoverflow.com/questions/14631776/calculate-turning-points-pivot-points-in-trajectory-path)

The RDP algorithm effectively works by drawing a straight line between the starting and ending points of the path, and finding the point on that route that is the maximum distance from that line segment and comparing it against a provided tolerance. If the largest point is outside that tolerance, is remembered as an RDP point, and the path is split and the RDP point serves as a new starting and ending point for the line segment.

Once I had those RDP points, I assigned every adjacent set of 3 points, and every set of 3 RDP points with one skipped between them, an angle and the distance of the legs of the triangle. I then matched every path against every other path and looked for at least 3 matching angles / leg distances that could be rotated to be in the same locations.

I was very surprised at the end of the competition to see that many teams utilized the RDP algorithm, since it never broke on the forums during the competition, although many other valuable insights did.

Andrei: Gradient boosting proved the reputation of performing well on a small training dataset. There is a perfect in-depth tutorial which I would recommend http://topepo.github.io/caret/index.html (http://topepo.github.io/caret/index.html) especially about model training and parameters tuning.



(http://blog.kaggle.com/wp-content/uploads/2015/04/axablog1.png)

SCHEMATIC ILLUSTRATION OF OUR MODELLING PROCESS. WE EXTRACTED 3 FEATURE SETS AND TRAINED DIFFERENT CLASSIFIERS ON THEM (RF: RANDOM FOREST, GBM: GRADIENT BOOSTING MACHINE, LOGREG: LOGISTIC REGRESSION). EACH TRIP WAS ASSIGNED A PROBABILITY (OF BEING A TRUE TRIP). IN EACH CLASSIFIER OUTPUT WE GLOBALLY SORTED THE TRIPS ACCORDING TO THOSE PROBABILITIES AND ASSIGNED RANKS (ACROSS DRIVERS). THESE WERE USED AS INPUT FOR THE FINAL ENSEMBLE. OUR FINAL TELEMATICS MODEL CONSISTED OF A WEIGHTED SUM ENSEMBLE OF THE 6 DIFFERENT MODELS.

What was your most important insight into the data?

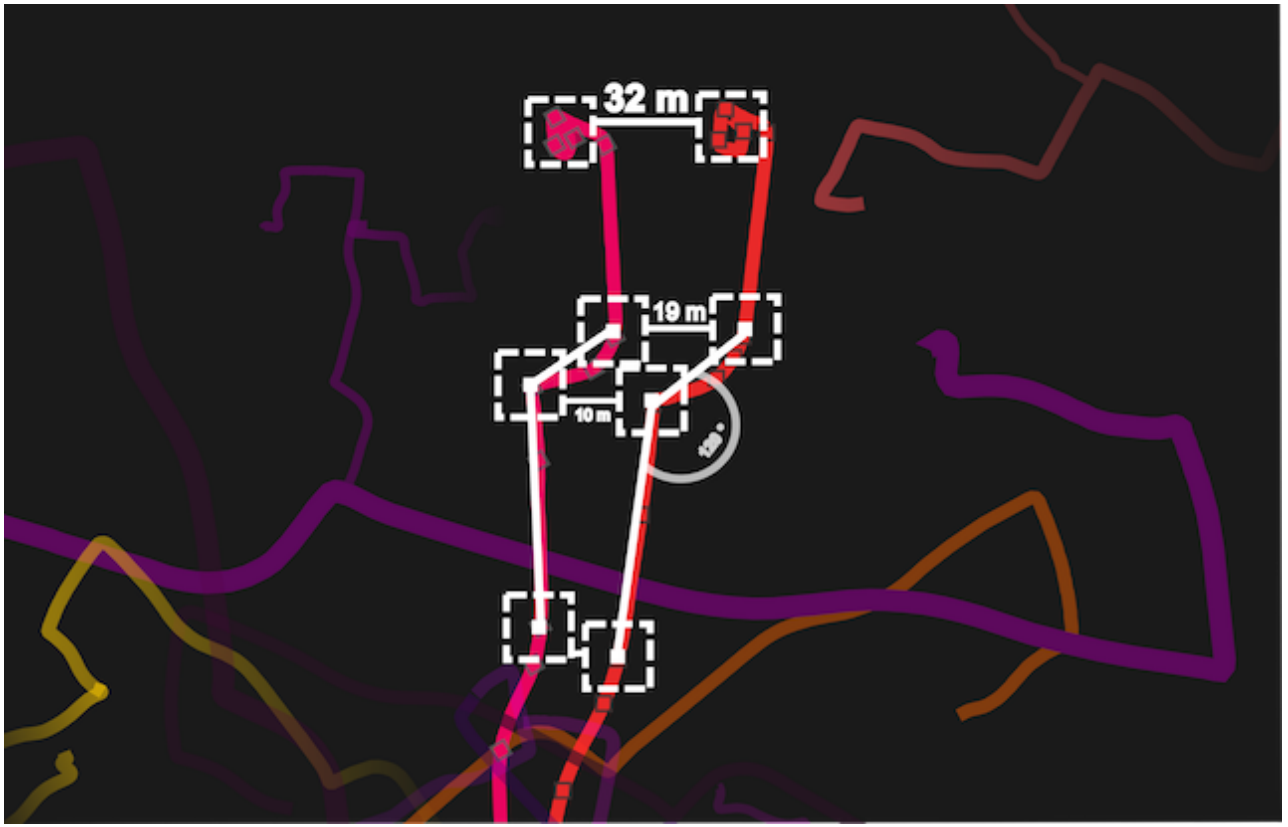
Scott: For me, it was the power of ensembling. Early on I thought that the competition could be won by path matching. However it quickly became apparent that the best would be combination of path matching and telematics. Since I was not familiar with the telematics approach, I found teammates who could do this. As soon as the team formed, combining my ~120th ranked path matching with my teammates ~120th ranked telematics, we immediately jumped up to 25th place.

Ensembling helped us at the end too. We had 6 different telematics solutions scoring between .83 - .89, and combined they scored .91. Basically, anytime we rolled in a independently scored telematics solution we were able to get at least some score boost.

Janto: I agree with Scott, the ensembling was key. Early in the competition it seemed there were two types of competitors: the "telematic modelling guys" and the "trip matchers", both obtained good results in the 0.9 ballpark. I soon realized that trip matching would be important to perform well in this task, so we got together.

For me, the main insight was the fact that the two distinct components of the task - geometry matching and driver modelling - could be best combined by using "probability ranks" for the trips. The probability distributions of the different models often looked quite different and the trip matching basically produced counts as output. So, we had to translate the probabilistic outputs and counts into one common metric.

Andrei: If we get to the origin of the challenge, namely identifying the driver fingerprint, the path matching part originally sounded like actually cracking the competition. I still think it does not contribute the fingerprint, but that was probably the only way to handle the fact, that real coordinates were anonymized and trimmed down. Having some kind of the “driving landscape”, i.e. types of driving areas, would be more useful feature in real life.



(http://blog.kaggle.com/wp-content/uploads/2015/04/axablog2.png)

TRIP MATCHING. THE TRIPS ARE DOWNSAMPLED USING THE RAMER-DOUGLAS-PEUCKER (RDP) ALGORITHM. SUBSEQUENTLY, OUR ALGORITHM SEARCHED EXHAUSTIVELY FOR TRIPLES OF GPS POINTS WITHIN A SPECIFIED DISTANCE THAT MATCHED IN ANGLE AND COULD THEREBY MATCH ROUTES BUT ALSO ROAD SEGMENTS. VARYING THE SEARCH DISTANCE, I.E. THE SENSITIVITY OF THE ALGORITHM, AND INTEGRATING THE RESULTS WE COULD SIGNIFICANTLY IMPROVE OUR TRIP MATCHING.

Were you surprised by any of your findings?

Scott: I was surprised at the number of junk runs there were. Approximately 3% of the data was just garbage that never left 100 meters from the starting point. We assumed that these was accidentally turned on GPS or similar. We ended up getting a little bit of AUC value out of them by simply counting the number of junk runs per each driver, and if a driver had a lot of junk runs, moving each of those runs higher in the rankings. (Some of the drivers had 30, 40 or more junk runs!)

Janto: My SOMs performed surprisingly bad.

Andrei: The “real life” nature of the data was indeed the most “uncomfortable” thing, that automatically gets you out of the comfort zone of working with the reference samples. There were many concerns about this in the forums as well.

Which tools did you use?

Scott: For the path matching I used mostly python, with numpy and matplotlib. The path matching was nice because the paths could be plotted for immediate feedback on if the algorithm was working or not. With the path matching, I perennially had a problem with run time. For most of the competition, my path matching algorithm took 2 days to get through all 2736 drivers on my 4 core computer. I spent substantial time improving the code to make it run faster, but that was quickly eaten up by making it more complicated to perform better. Eventually, I wrote some of the more simple, but computationally long parts of the algorithm in Fortran, which processes For loops approximately 100 times faster than python, and used F2Py to tie them into the python.

At the end of the competition, we needed more computing power, so I launched 7 two node virtual machines on google compute, as well as another on Azure for a total of 16 nodes running for 14 days. (The cost of this would have been on the order or ~ $100, however Google has a $300 free credit which I utilized.)

For the telematics, we utilized Random Forests and Gradient Based Trees from the Python SciKit Learn toolkit.

Janto: I ran all my analyses in Python, using the great numpy module and scikit learn toolkit for the modelling. For visualizations I used matplotlib. We collaborated via GitHub. As our repository was overflowing with scripts combining those in batch scripts was also quite helpful.

Andrei: I mainly used R with the *caret* package (http://topepo.github.io/caret/index.html (http://topepo.github.io/caret/index.html)) Azure cluster was used with R Enterprise from http://www.revolutionanalytics.com/revolution-r-enterprise (http://www.revolutionanalytics.com/revolution-r-enterprise) and *doParallel* backend http://cran.r-project.org/web/packages/doParallel/vignettes/gettingstartedParallel.pdf (http://cran.r-project.org/web/packages/doParallel/vignettes/gettingstartedParallel.pdf)

What have you taken away from this competition?

Janto: Collaborating in a team is a great experience, we finished 2nd place because we joined forces. I definitely recommend it. Also, we organized our code on Github which I had never used before and it’s a great resource when working together. The competition taught me the power of ensembles and how knowing the evaluation metric can really help to make informed choices in

the ensembling process.

Scott: The value in utilizing Github when working with teammates. The basics of using the sci-kit learn toolkit, especially random forests and gradient boosted trees for data analysis. The value of teams, since I would certainly not have scored as highly working on my own.

Andrei: One does not need to be a domain expert to already perform above average. I do not think we really innovated in the feature development, neither we did in the methods. As it was indicated above, many competitors tried the same. But mainly focusing on different machine learning techniques and organized them together helped us to boost the performance. During last week the process was pretty much “wait 4 days to complete” to get 0.0005 performance.

Do you have any advice for those just getting started in data science?

Janto: In my opinion, one important point is to be able to test ideas as efficient and reproducible as possible. I like to build pipelines for every necessary task, like "preprocessing","modelling", "validation" and "submission" from the beginning. These modules can be easily scaled up at later stages and you don't have to care anymore about the code that produces your predictions if you change your model.

It helps to have one script in which you only specify the model or the ensemble of models, one that loads the data, one that preprocesses the data, etc. If you now wake up in the middle of the night and think "I should really change my logistic regression to an ensemble of random forests and a gradient boosting machine and change the window length of my moving average preprocessing to 10s" you just change two lines in the model and your preprocessing script and type "python validate.py" in your shell. This helps to keep an overview over your code and avoids errors in testing or output routines. And you can test a new ensemble when you come home from a party at 4 am before you go to sleep.

Validate your models and keep a log. In this competition it turned out that public and private scores correlated nicely but you can’t rely on that. Keep track of every cross-validation and the corresponding LB score.

In this competition it was a bit tricky to validate model performance since the data was unlabeled. That's why I transformed the problem into a supervised one and recreated the task, that is, detecting outliers, with known labels: I took the 200 trips of one driver as "real" trips and injected k (usually around 5-10) artificial false trips from another driver into the dataset. Now I had labels and could run a classification scheme. Doing this for around 100 drivers and computing the global AUC score yielded a good indicator of performance.

We observed that the public scores were really close to the local CV scores. This observation justified that we trusted the leaderboard score for some time consuming complex analyses that took a couple of days to compute at later stages of the challenge.

Scott: Watch the forums in your competition. There are a ton of insights to be had in there. Our team got a lot of the features that we ended up using for the telematics competition based on posts from the forums. And our basic method of path matching combined with telematics was all over the forums

Really get to know what the scoring criteria is. The AUC criteria used for this competition essentially only cares about the order the results are ranked in, not the actual scores. We got a large score boost (~.005) by making sure that no two results were scored the same by sorting our path matching results by the telematics scores and doing a few other tricks to make sure that each of the scores would be unique.

Also, don't bother with running super computationally expensive runs early on. You can always crank up the number of trees in your random forest for the final few submissions, early on it is valuable to be able to iterate quickly. If I were to do this competition again I would have spent less time running the full 2736 driver data set in January / February and run more iterations of just 10 or 100 drivers that I could check and improve quickly.

Andrei: Use things like google scholar to find few relevant research papers. Especially if you are not a domain expert. Some domains have a better coverage of different machine learning techniques then the others. It is not worth inventing things until the proven methods have not been tried.

The Team

Andrei Varanovich (http://blog.kaggle.com/wp-content/uploads/2015/04/andrei.jpeg) With a diploma in System Engineering and Masters in Computer Science, Andrei is currently finishing a PhD thesis as a result of 4 years spent as part of the Software Languages Team at the University of Koblenz-Landau in Germany. In his role as cloud architect at Sparked, he mainly focuses on data analytics and machine learning in the Microsoft ecosystem. He is a Microsoft Most Valuable professional since 2008.

Janto Oellrich is a Cognitive Science MSc student specializing in Neuroscience and Machine Learning. Currently, he is working as a research assistant at the Department of Experimental Psychology at the University of Oxford. His research interests include perceptual decision making and feature representation in humans, computational modelling of behaviour and machine learning.

Scott Hartshorn is an aerospace engineer and a graduate of the University of Michigan. His professional work has been in structural engineering. He is currently working to develop an expertise in Machine Learning and Data Analytics.

AXA DRIVER TELEMATICS ANALYSIS (HTTP://BLOG.KAGGLE.COM/TAG/AXA-DRIVER-TELEMATICS-ANALYSIS/)