Assessment 1 Reflection

Tom Blain

For the project we decided to work on Porto Seguro's Safe Driver Prediction, a past Kaggle competition. We chose this option because it contained a difficult to work with big dataset, and allowed experimentation with a lot of methods. The dataset posed multiple challenges: missing values, unhelpful feature names, unbalanced classes, and anonymized data as is common practice within the insurance industry.

We each had different models in mind when approaching the problem. In reflection, many of the conclusions we reached could have been predicted by a more thorough EDA stage. However, implementing the models and working with the team to solve issues was a great learning experience. EDA would have allowed us to see that ultimately, there was a complex non-linear relationship between the features and the target, putting a limit on how well a lot of models would have been able to perform. It was insightful at this stage to read up on the problem of imbalanced classes and how we can use stratified methods for CV.

Logistic regression worked as a baseline model, however, parameter improvements and data manipulation did not allow the model to perform close to that of Boosting and I would assume random forests would also outperform. I think our work into the KNN model was flawed because the advantages and disadvantages of KNN were not fully understood - the model was fed around 60 features of not great data, so in reflection, it is unsurprising that our model produced results that were only slightly better than random guessing. Maybe there are ways of making this approach perform better if we were to use RF importance of features or PCA in the future to carefully select the data we give KNN. There also was not any study done into whether alternative distance metrics would work better.

It was good practice working as a team to study what we can do about the large amount of missing values, and comparing model performance with mean imputation compared to a regression imputation. In the future I would like to study some more advanced methods such as ANN/DNN imputation, GAN's, clustering, or multiple imputation. After speaking to data scientists, it seems like they will almost never have a perfect dataset to work on and this is a common challenge.

I set about implementing the boosting algorithm starting off with XGboost. Boosting does not usually require encoding or scaling but I implemented these anyway since the other models would require them and it is unlikely to harm the model. I found it very simple to set up a model initially realising that improvements would come from hyper-parameter optimisation and tuning the data. Literature online was instrumental since I had not worked with boosting algorithms previously. My main area of studies when working with boosting

was how to best avoid overfitting which introduced me to k-fold CV, and making sure my training choices were justifiable and not simply based on what would give me the best score - such as stopping times and learning rates.

Hyper-parameter optimisation was one of the biggest learning experiences for me in this task. Bayesian optimisation made sense to me as a much more efficient algorithm but I was initially stuck trying to implement it so I turned to grid search since it was built into Sk-learn. I'm not sure why literature online chose grid search and not random grid search, as further reading of papers in this topic concludes that randomised is almost always better. Grid search was too slow to run - to check 40 different parameter combinations on k=3 folds, it would take hours. It was unsatisfying because I knew even after running the code for hours, if I was able to speed up the algorithm we could check a lot more combinations and have a more successful model. I tried reducing the data size for the grid search but it was still relatively slow even at 25%. I had heard about LightGBM before and have seen that this is an approach a lot of top performing kaggle submissions choose. Instantly the speed was about 10x as fast.

I decided to continue reading literature online knowing that bayesian methods for optimisation would still be better in almost every way - bayesian optimisation allows the hyper-parameter search to focus around the 'optimum' value much more quickly and waste less time looking at bad combinations. Furthermore, it can find continuous values of our hyper-parameters whereas grid search was limited to the values put into the grid. I am pleased with learning how to use bayesian optimisation libraries in python as I see this as something I would always try to use in the future when possible.

I feel I went above and beyond in this project to support the team. As the most experienced at ML and python, I made sure to try and always be available to answer questions and contribute heavily in group discussions. I realise that a lot of my learning in this area has been due to working with others who have more experience with ML implementations than me. Daniel and Xinnyue had a great theoretical background in the statistical parts of the project and were very motivated to better their coding knowledge.

My overall contribution to the project was the data handling section, boosting algorithm, the conclusion and part of the evaluation explaining the Gini score. I feel I led the team with organisation and our approach to the task.

In conclusion, the project was a great learning experience for me with the main takeaways being the effectiveness and implementation of high performing ensemble models, how to deal with the difficulties we faced in the data such as missing values and class imbalance, and exploring a lot of DS techniques to overcome challenges I hadn't faced before.