**Quiz Questions Response**

**24 February 2019**

**Darren Wilkinson**

**Q1. Short outline of approach to the task:**

* Import data, aligning train & test data sets.
* Objective for first pass is to get simple benchmark with minimal data cleaning, feature selection and data analytics. This would typically represent a significant amount of value when the features are known.
* Feature engineering and dimensionality reduction no performed.
* Feature set was scaled and “id” values removed.
* Split the approach into 2 core models to help accuracy given the relatively low number of defaults to non-defaults:
  + Default Classifier (binary 1 or 0)
  + Default value calculator.
* Random forests are generally a solid benchmark for both classifiers and regressions with a significant number of features, and tend to have fewer hyper parameter considerations. Clearly the model investigation could be expanded to:
  + GBM or neural nets for classifiers.
  + Multilinear regression for regression.
  + Deep learning models across the whole process (classifier & regression).
* No sample weighting (default vs non-default sample size rebalancing, eg SMOTE) or cross validation in the first iteration. This would be done as a follow-up activity.
* Simple binary prediction accuracy chosen as initial target metric for classifier.
* Mean Absolute Error (MAE) for regression to calculate load default value. This could be expanded to F1 or AUC metric for the classifier at a later stage once sensitivities are better understood. It could also be expanded to grid value optimisation for F1 and cut-off values to feed into overall MAE minimisation model at a later stage.
* Cases that the classifier predicted as default are passed to the regression model to calculate the loss value. Cases that are not predicted to default are expected to have zero loss.
* All run on a Jupyter notebook using Python 2.7.
* Set data location and save location in second box.
* Code is first iteration, ie latest code & data manipulation not implement, warnings not supressed, very little modularisation, etc.
* Predictions of test set default values written to a “preds.csv” file.
* Accuracy levels between train and cross validation set <5% difference for the accuracy classifier & MAE.

**Q2. Use and importance of PCA for dimensionality reduction.**

* PCA is a methodology to reduce the number of features in a dataset. It is done through creating linear combinations of features so that their information is maximised (ie explain the most variance). The number of features ultimately required is reduced.
* The first principle component is the one that explains most of the variation with the dependent variable.
* Gains in feature set efficiency are offset through interpretability, ie it is often hard to attribute meaning principle components.
* PCA is a popular approach in cluster analysis.
* It is also useful when the number of features is high relative to the number of cases.

**Q3: Explain how the model would run in the cloud.**

This depends on the speed of data ingress (is real time streaming data solution needed), number of users, frequency of update, data security / pseudonymisation, etc.

In Microsoft Azure this could be:

* Ingest: Azure Data Factory
* Store: Blob storage
* Prepare & Train: Databricks using Python (possibly Tensorflow & R) and maybe Spark depending on the volume and frequency of the data.
* Deployed model: container instance on Azure that can be accessed by Webapps and outputs into analytics services such as PowerBI, Tableau possibly via SQL Data Warehouse.

***Q4. Considerations if larger dataset provided***

* Increase processing capacity, RAM and parallel compute ability (eg map / reduce capability with Spark clusters)
* Use batch processing of algos
* Minimise loops and optimise processing speeds across different libraries.

***Q5. Changes to code to run in production environment***

* More modularised
* Containerised
* Add test conditions
* Ensure more robust (to odd data types, missing data or access problems) and build in fall over procedures
* Build and deploy code using code repositories with roll-back functionality