Telstra Project

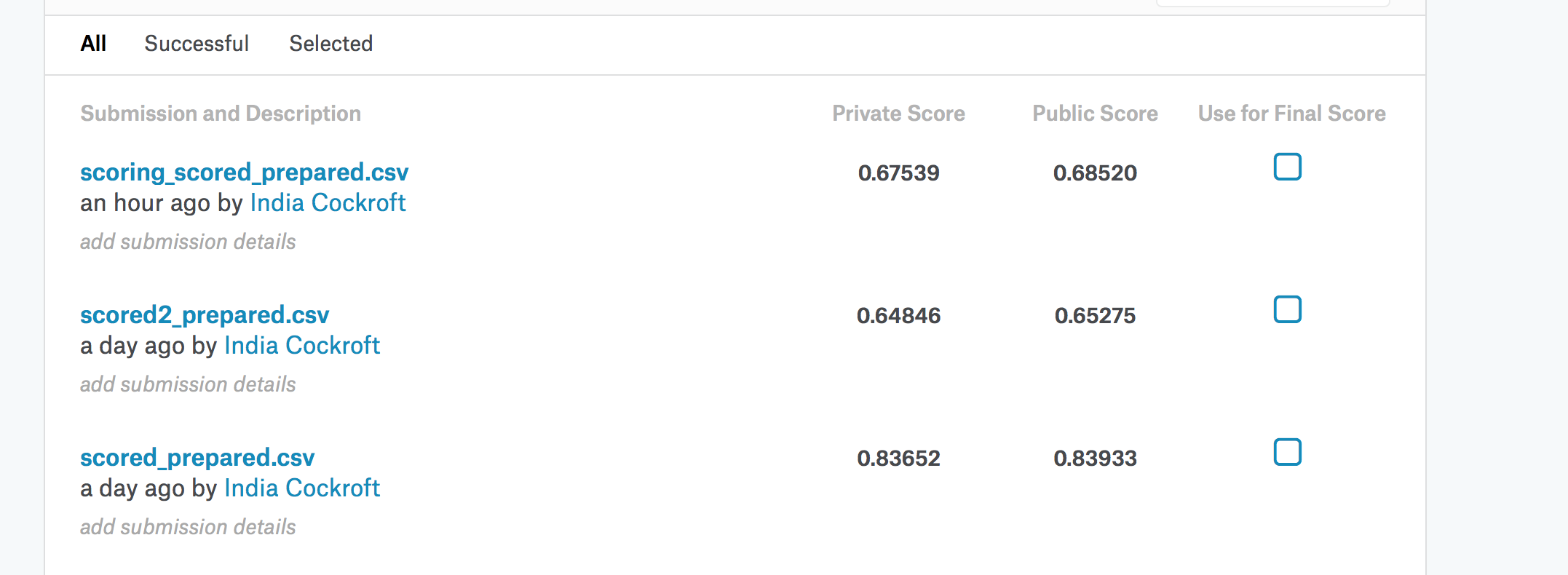
Please note that for this Telstra Project, I worked with Tomas as he said that you and him had been in contact and you were very helpful explaining the correct method.

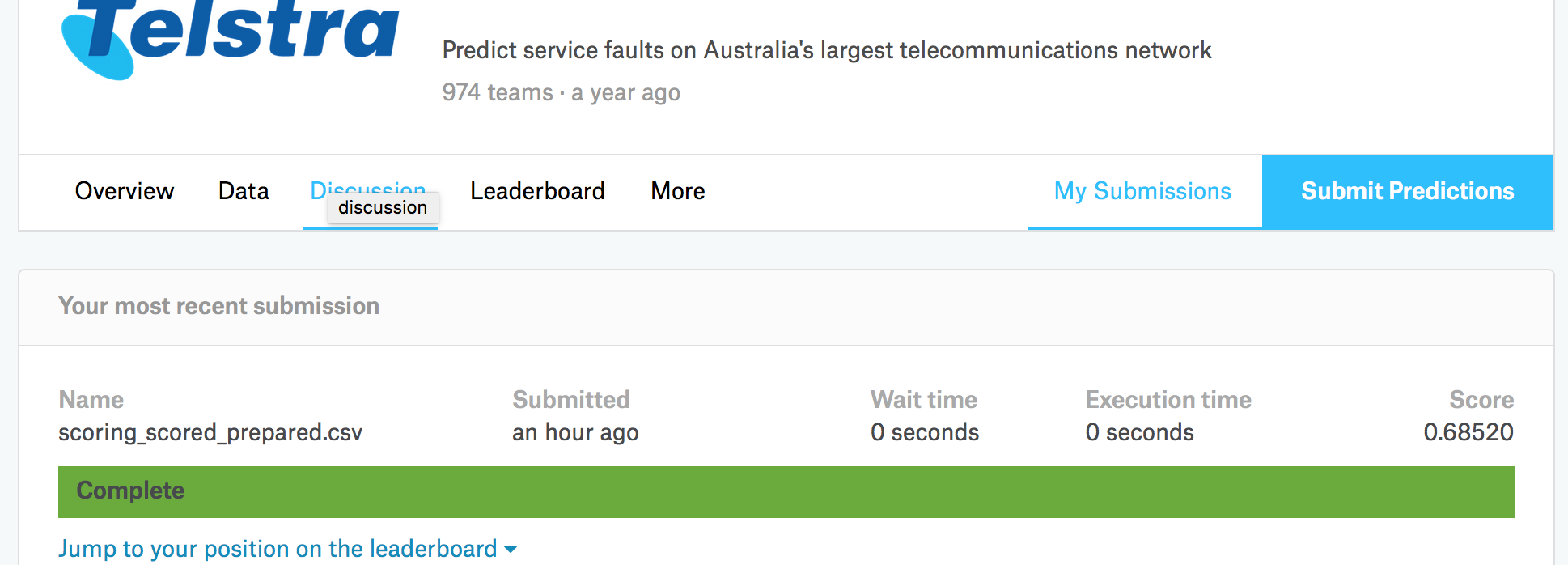
For each of the initial datasets, I synced these to create a copy that was storied in Postgres. Then for the log\_feature\_copy , resource\_type\_copy and event\_type\_copy datasets, an SQL code recipe was used to create count columns and string aggregations for the corresponding features and the data was grouped by id.

For the test\_copy and train\_copy datasets, these were stacked and then joined with the severity\_type\_copy by id. The names of the columns were changed in a lab analysis to create severity\_type\_copy\_prepared\_prepared2.

Then the 4 datasets were all joined using a left join by id and run in DSS to created information\_joined. From here the data was split by the fault\_severity column into 2 sets of data, ‘training’ and ‘scoring’ where the values 0, 1 and 2 were assigned to the training set and the rest to scoring.

A random forest model was created for the training set. When producing the models, random forest and XGBoost were used for the algorithms. The most important variables were noted – the leading variable being location followed by severity type. This model was then scored to the scoring dataset to produce ‘scored. Following this, the data was prepared so it was in the submission format for kaggle. The results and submissions are shown below.

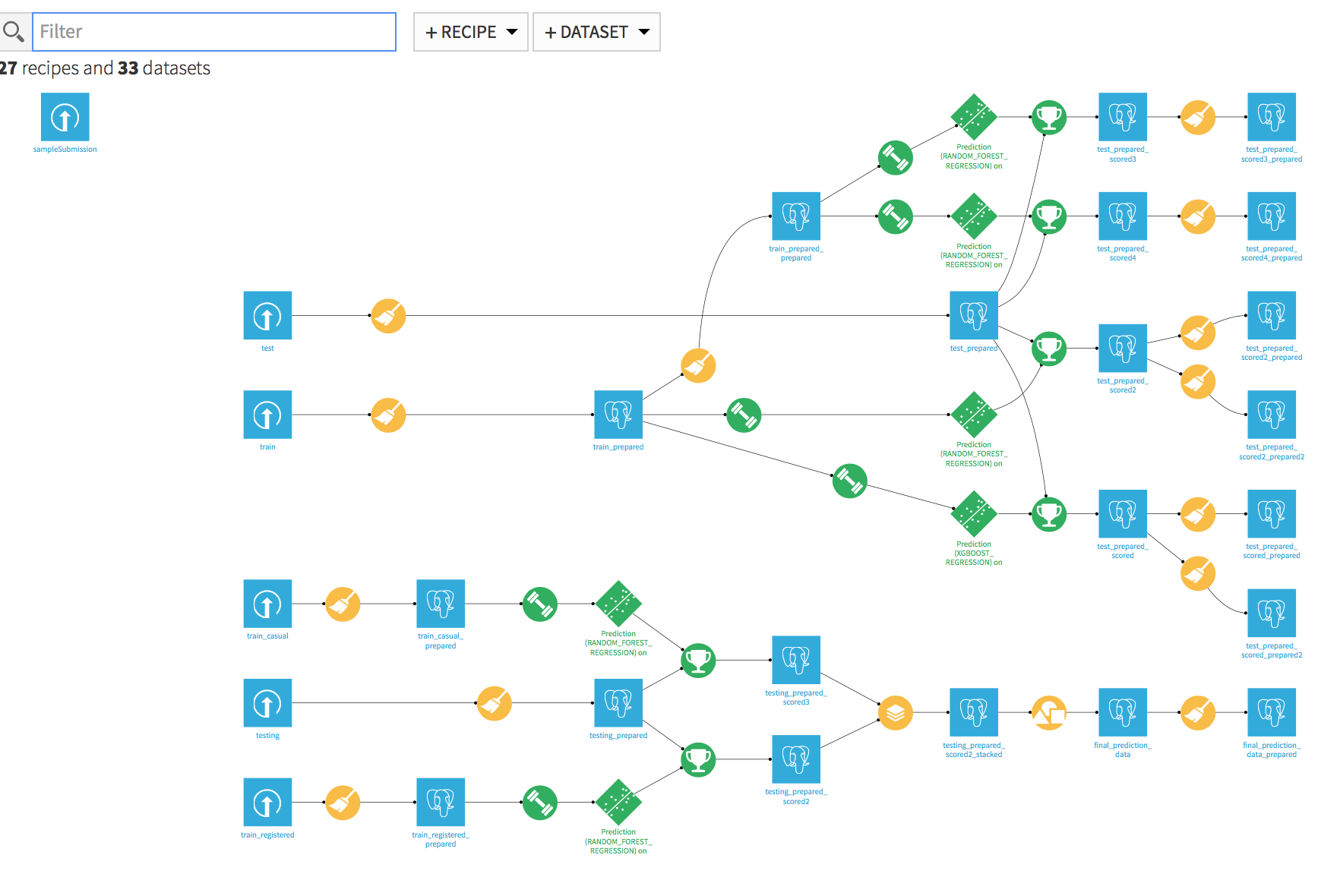




Bike Sharing Demand Project

Bike sharing systems are a means of renting bicycles where the process of rental, return an membership is through a network of kiosk locations throughout a city. The data generated from these systems is attractive for researched because the duration of travel, departure location, arrival location and time is recorded. In this project, historical usage patterns in weather data must be combined to forecast bike rental demand in the Capital Bikeshare program in Washington DC. The hourly rental data from the past two years has been provided in form of two datasets- The training dataset contains information regarding the first 19 days of each month, while the test is only the 20th to the end of the month. The total count of bikes rented during each hour covered by the test dataset must be predicted, using only the information available prior to the renal period.

The flow of my project is shown below.



My initial method is shown at the top of the flow. I prepared the test and train datasets and then created various models for the train set before scoring them to the prepared test data. I then prepared the data for submission. However, I realized that I was not taking into account the registered and casual user counts using this method (the train data contains three extra columns outlining the registered user count, the casual user count and the overall count).

Therefore, I changed the method to try improve my predictions as shown in the bottom of the flow photo. The train data was uploaded twice, one called train\_casual and one called train\_registered and the test data was also uploaded. To prepare the data for the train\_casual, I parsed the datetime and extracted the month, the day of the week and the hour before deleting the count, the registered, and datetime\_parsed column. The casual column was then renamed as casual count. The same process was carried out to prepare the train\_registered dataset except the casual users column was deleted. Also the types of data was focused on and it was made sure that the data types were the same in both datasets – for each column the type was ‘bigint’ defined as an Integer with the exception of the datetime, which was of string type and the temp, atemp and windspeed columns which were ‘double’ data defined as a decimal.

The data types were also the same for the test data. To prepare the test data, the date was parsed before the same components were extracted and datetime\_parsed column deleted. Two separate models were then computed using the Random Forest and XGboost algorithms. The XGBoost algorithms created negative predictions – I changed the negative numbers into zeros to then submit the predictions but these had worse results than the random forest. To create the models, the performance template was used.

The two models were then scored to the testing\_prepared data and the results stacked so that the casual user prediction and the registered user predictions were shown for each datetime. Then the data was grouped by datetime and the predictions were summed to add together both the counts. The prediction\_sum column was changed to ‘count’ and the data prepared for submission. My submission results are outlined below showing the best result which was for this second method.

